US006594682B2

(12) **United States Patent**
Peterson et al.

(10) **Patent No.: US 6,594,682 B2**
(45) **Date of Patent: Jul. 15, 2003**

(54) **CLIENT-SIDE SYSTEM FOR SCHEDULING DELIVERY OF WEB CONTENT AND LOCALLY MANAGING THE WEB CONTENT**

(75) Inventors: **Leonard J. Peterson**, Woodinville, WA (US); **Steven J. Freedman**, Seattle, WA (US); **Hadi Partovi**, Seattle, WA (US); **Raymond E. Endres**, Kirkland, WA (US); **David J. D'Souza**, Kirkland, WA (US); **Erik Castedo Ellerman**, Redmond, WA (US); **Julian P. Jiggins**, Seattle, WA (US)

(73) Assignee: **Microsoft Corporation**, Redmond, WA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **08/959,613**

(22) Filed: **Oct. 28, 1997**

(65) **Prior Publication Data**

US 2001/0003828 A1 Jun. 14, 2001

(51) **Int. Cl.**[7] **G06F 15/16**
(52) **U.S. Cl.** **709/102**; 709/219; 709/232
(58) **Field of Search** 709/277, 228, 709/229, 232, 233, 243, 202, 203, 102, 200, 217, 218, 234; 713/600

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,715,443	A	*	2/1998	Yanagibara et al.
5,754,939	A	*	5/1998	Herz et al. 455/4.2
5,758,257	A	*	5/1998	Herz et al.
5,760,771	A	*	6/1998	Blonder et al.
5,768,528	A	*	6/1998	Stumm 395/200.61
5,778,187	A	*	7/1998	Monteiro et al.
5,790,790	A	*	8/1998	Smith et al.
5,832,223	A	*	11/1998	Hara et al. 395/200.47
5,832,232	A	*	11/1998	Danneels
5,832,496	A	*	11/1998	Anand et al.

(List continued on next page.)

OTHER PUBLICATIONS

Yahoo, "Internet Search Engine", http://www.yahoo.com, 2 pages. Printed May 10, 2002.
Lycos, "Internet Search Engine", http://www.lycos.com, 1 page. Printed May 10, 2002.

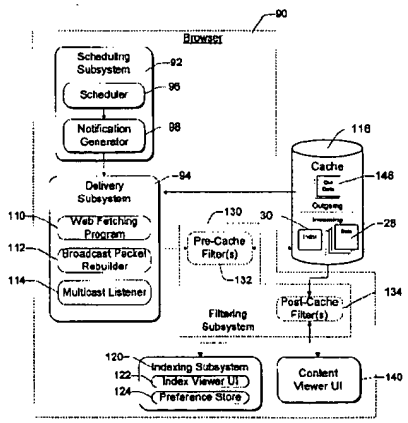
(List continued on next page.)

Primary Examiner—Le Hien Luu
Assistant Examiner—Stephan Willett
(74) *Attorney, Agent, or Firm*—Lee & Hayes, PLLC

(57) **ABSTRACT**

A client-based system has a scheduling subsystem to schedule a time to obtain the Web content from the server. When the client reaches the scheduled time, the scheduling subsystem generates an event notification that contains sufficient information explaining how to retrieve the Web content. The client-based system has a delivery subsystem that is responsive to the event notification to obtain the Web content at the time set by the scheduling subsystem. The delivery subsystem preferably has multiple delivery modules that enable different types of distribution mechanism. In addition to the Web content or data itself, the delivery subsystem obtains an index to the Web content. The index summarizes the Web content to facilitate local search and find tasks. The index and Web content are stored in a cache at the client. An indexing subsystem presents the index to a user and enables the user to select from the index portions of the Web content that they prefer. Based on these preferences, filters are created to remove items not of interest. When the client is offline, the user browses the cached Web content. The user is offered essentially the same functionality as a live online session, except that any requests to a remote server are temporarily accumulated for later submission. When the client reconnects to the server, all accumulated data is sent in batch to the appropriate servers. The user can also create his/her own channel by aggregating content from different channels.

26 Claims, 9 Drawing Sheets



U.S. PATENT DOCUMENTS

5,848,396	A *	12/1998	Gerace	
5,854,901	A *	12/1998	Cole et al.	
5,870,558	A *	2/1999	Branton, Jr. et al.	
5,907,681	A *	5/1999	Bates et al.	709/228
5,961,602	A *	10/1999	Thompson et al.	709/229
5,978,381	A *	11/1999	Perlman et al.	
5,978,842	A *	11/1999	Noble et al.	
5,991,306	A *	11/1999	Burns et al.	
5,999,664	A *	12/1999	Mahoney et al.	
6,065,059	A *	5/2000	Shieh et al.	709/233
6,134,584	A *	10/2000	Chang et al.	
6,182,113	B1 *	1/2001	Narayanaswami	709/203
6,275,496	B1 *	8/2001	Burns et al.	709/217
6,298,373	B1 *	10/2001	Burns et al.	709/203
6,324,182	B1 *	11/2001	Burns et al.	
6,442,598	B1 *	8/2002	Wright et al.	709/217
2001/0003828	A1 *	6/2001	Peterson et al.	709/219

OTHER PUBLICATIONS

Google, "Internet Search Engine", http://www.google.com, 1 page. Printed May 10, 2002.

Altavista, "Internet Search Engine", http://alta-vista.com, 1 page. Printed May 10, 2002.

GO.com, "Internet Search Engine", http://infoseek.go.com, 2 pages. Printed May 10, 2002.

* cited by examiner

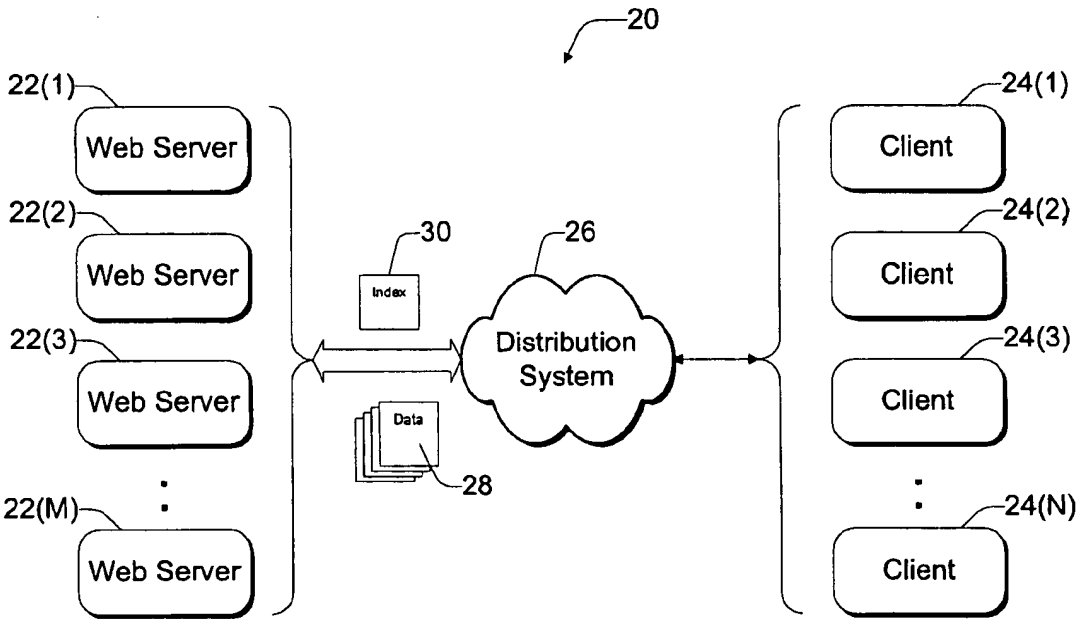

20
22(1)
Web Server
22(2)
Web Server
22(3)
Web Server
22(M)
Web Server
30
Index
26
Distribution System
Data
28
24(1)
Client
24(2)
Client
24(3)
Client
24(N)
Client

Fig. 1

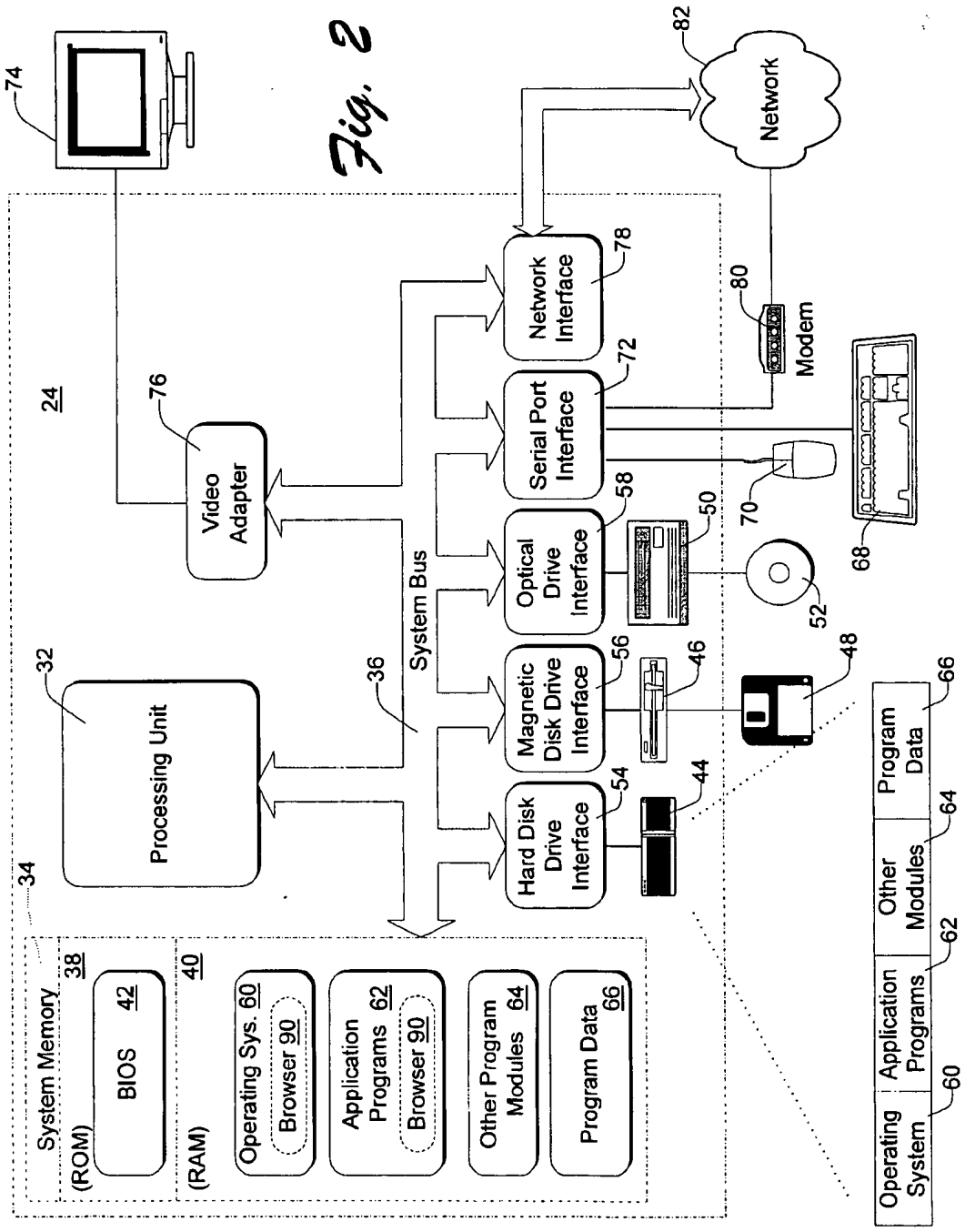


Fig. 2
74
82
Network
24
76
Video Adapter
Network Interface
78
Serial Port Interface
72
80
Modem
System Bus
Optical Drive Interface
58
50
70
52
68
32
Processing Unit
36
Magnetic Disk Drive Interface
56
46
48
Hard Disk Drive Interface
54
44
34
System Memory
38
(ROM)
BIOS 42
40
(RAM)
Operating Sys. 60
Browser 90
Application Programs 62
Browser 90
Other Program Modules 64
Program Data 66
Operating System
Application Programs
Other Modules
Program Data
60
62
64
66

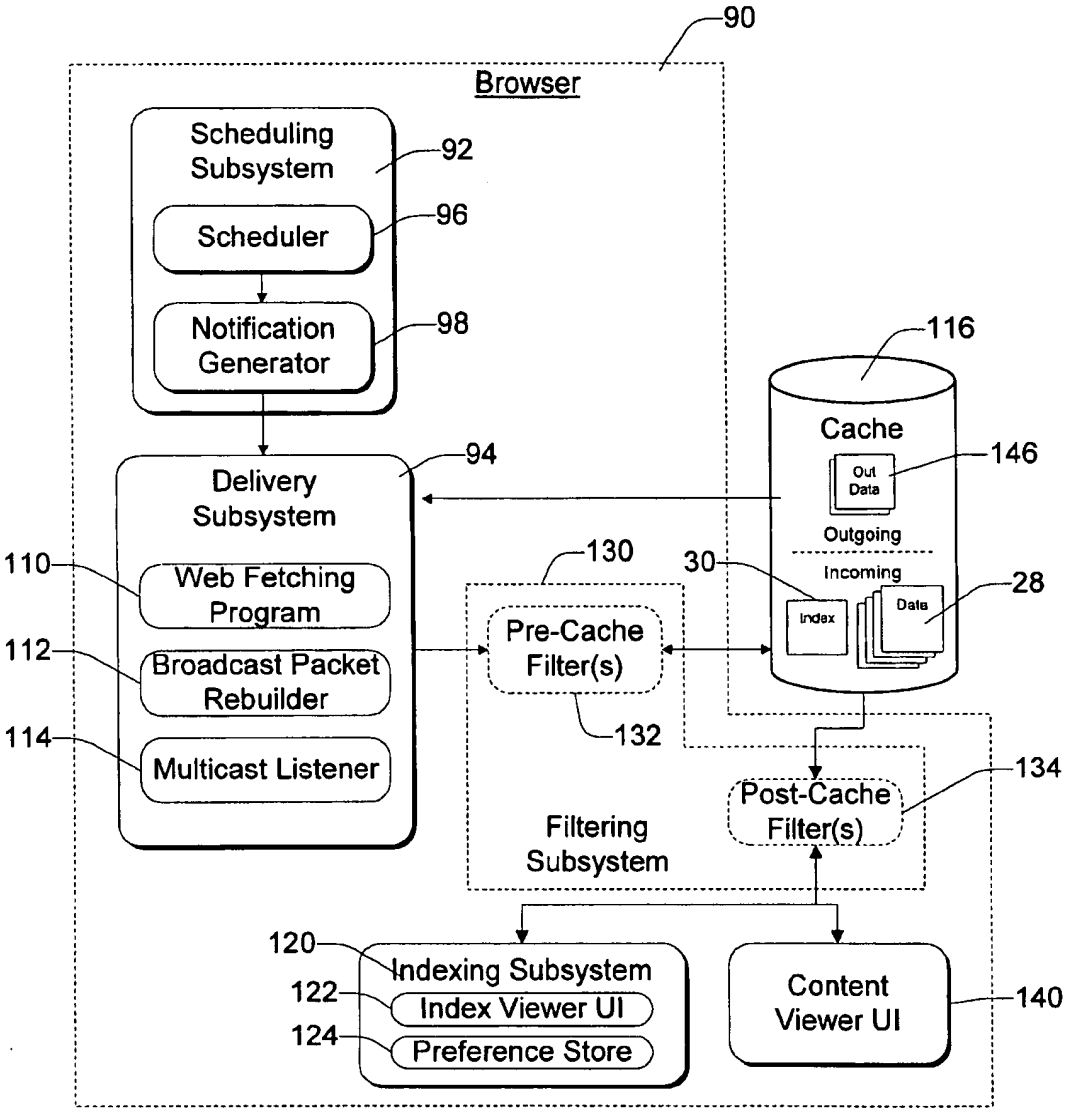

90
Browser
Scheduling Subsystem
92
Scheduler
96
Notification Generator
98
116
Cache
146
Out Data
Outgoing
Incoming
Index
Data
28
94
Delivery Subsystem
110
Web Fetching Program
112
Broadcast Packet Rebuilder
114
Multicast Listener
130
30
Pre-Cache Filter(s)
132
Post-Cache Filter(s)
134
Filtering Subsystem
120
Indexing Subsystem
122
Index Viewer UI
124
Preference Store
Content Viewer UI
140

Fig. 3

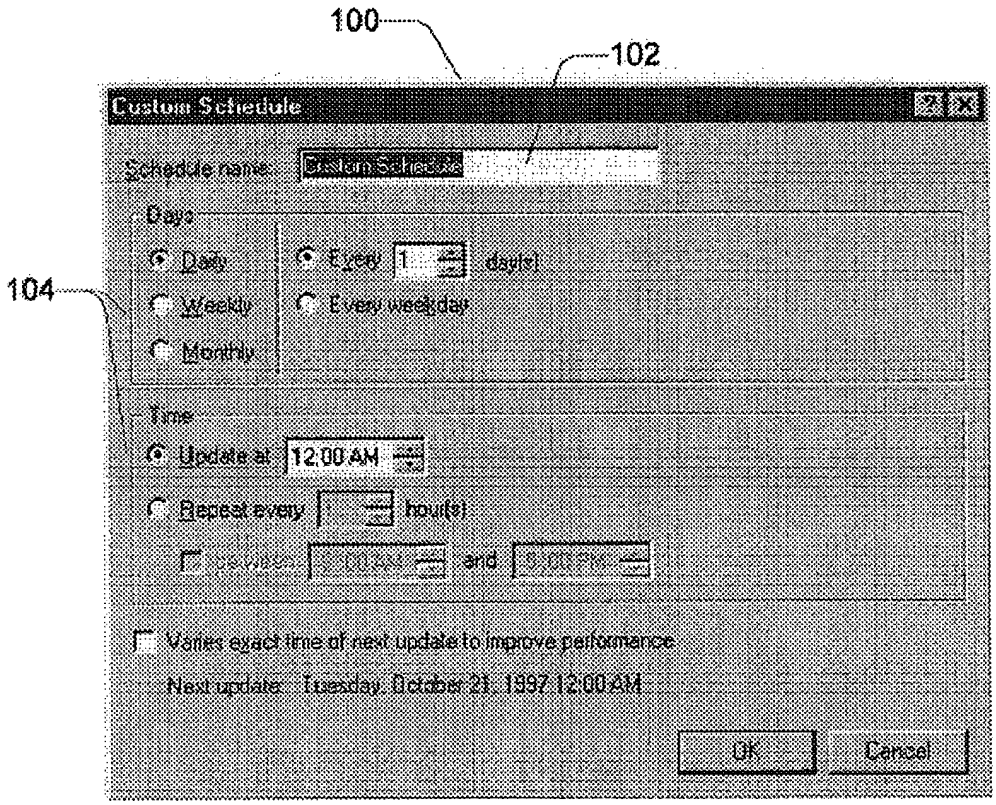

100
102
104
Custom Schedule
Schedule name:
Days
Daily
Every 1 day(s)
Weekly
Every weekday
Monthly
Time
Update at 12:00 AM
Repeat every hour(s)
and
Varies exact time of next update to improve performance
Next update: Tuesday, October 21, 1997 12:00 AM
OK
Cancel

Fig. 4

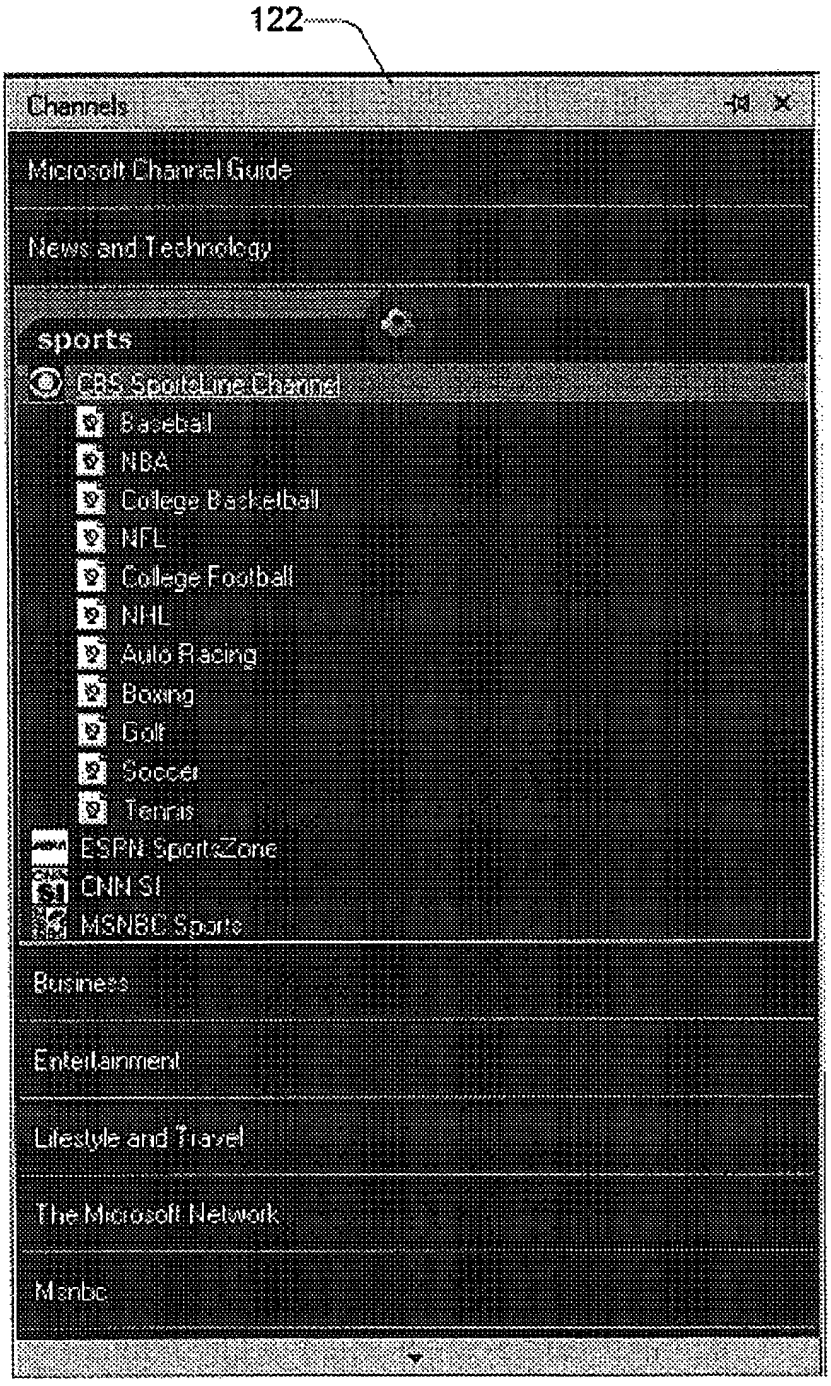


Fig. 5

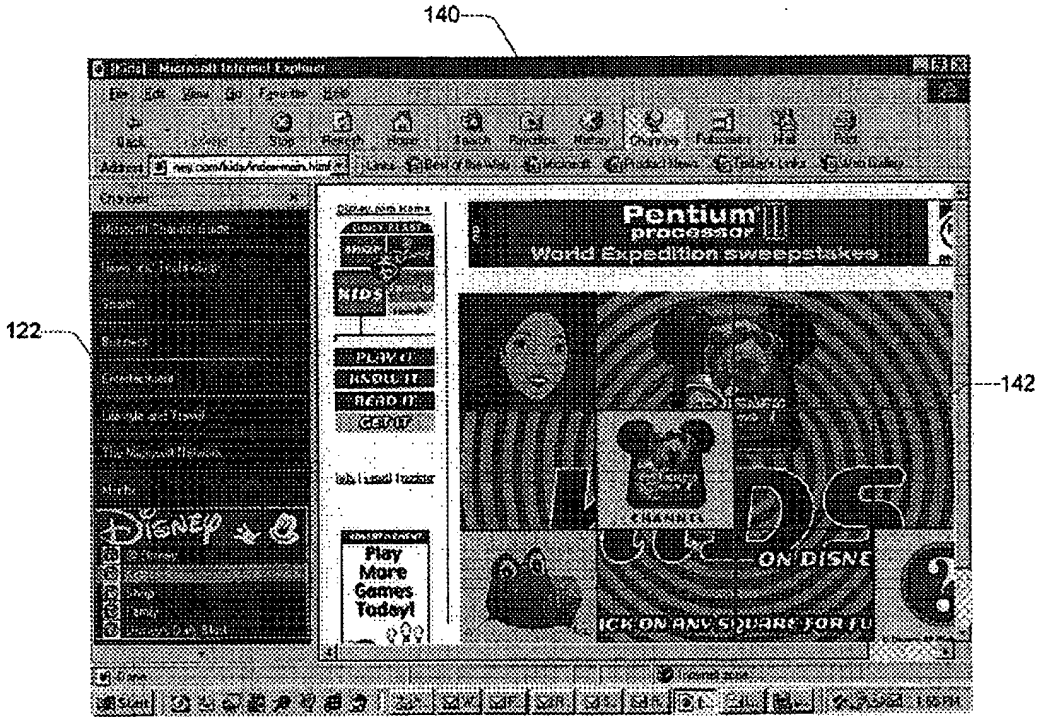


Fig. 6

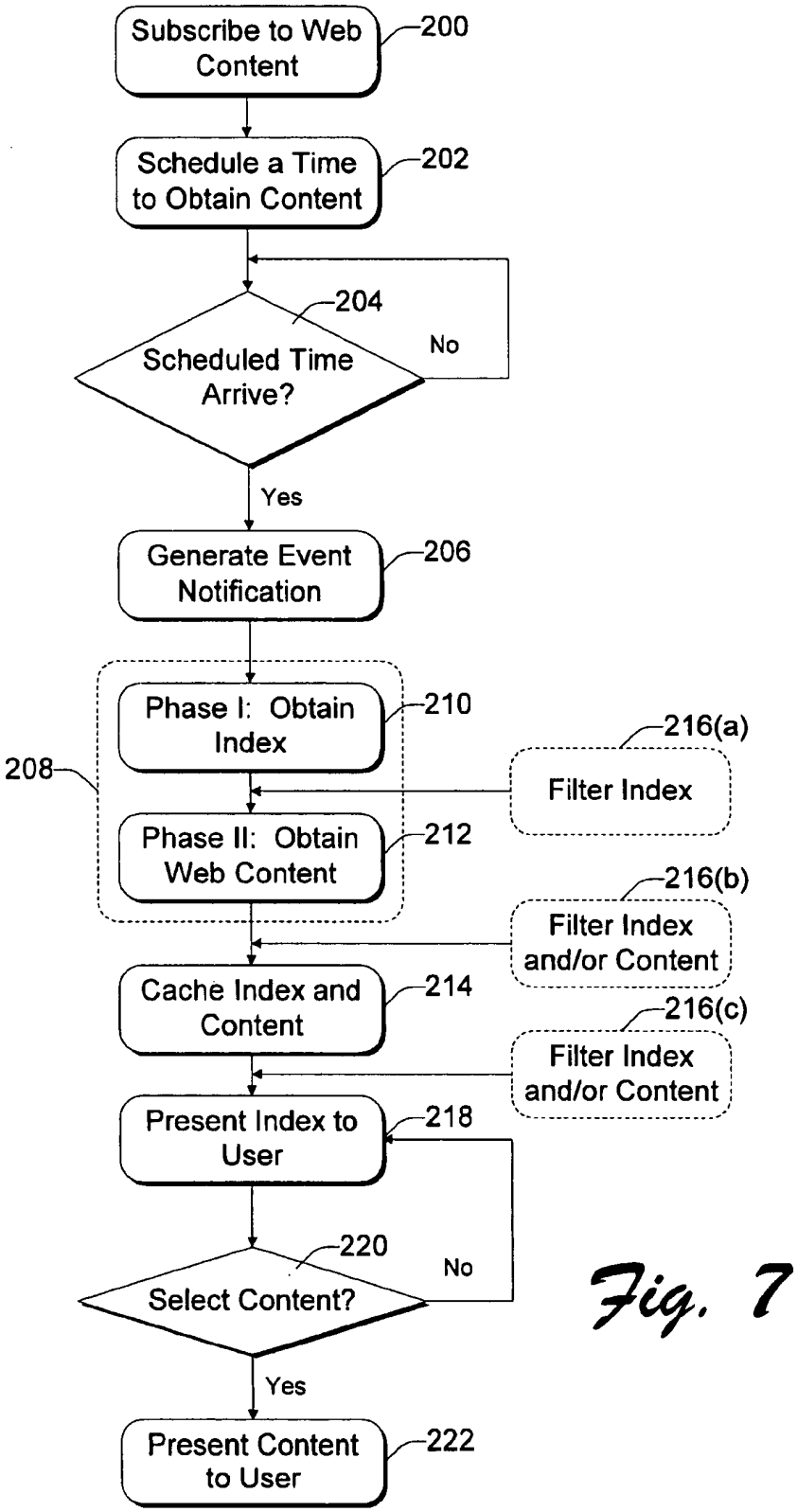

Subscribe to Web Content
200
Schedule a Time to Obtain Content
202
204
Scheduled Time Arrive?
No
Yes
Generate Event Notification
206
208
Phase I: Obtain Index
210
216(a)
Filter Index
Phase II: Obtain Web Content
212
216(b)
Filter Index and/or Content
Cache Index and Content
214
216(c)
Filter Index and/or Content
Present Index to User
218
220
Select Content?
No
Yes
Present Content to User
222

Fig. 7

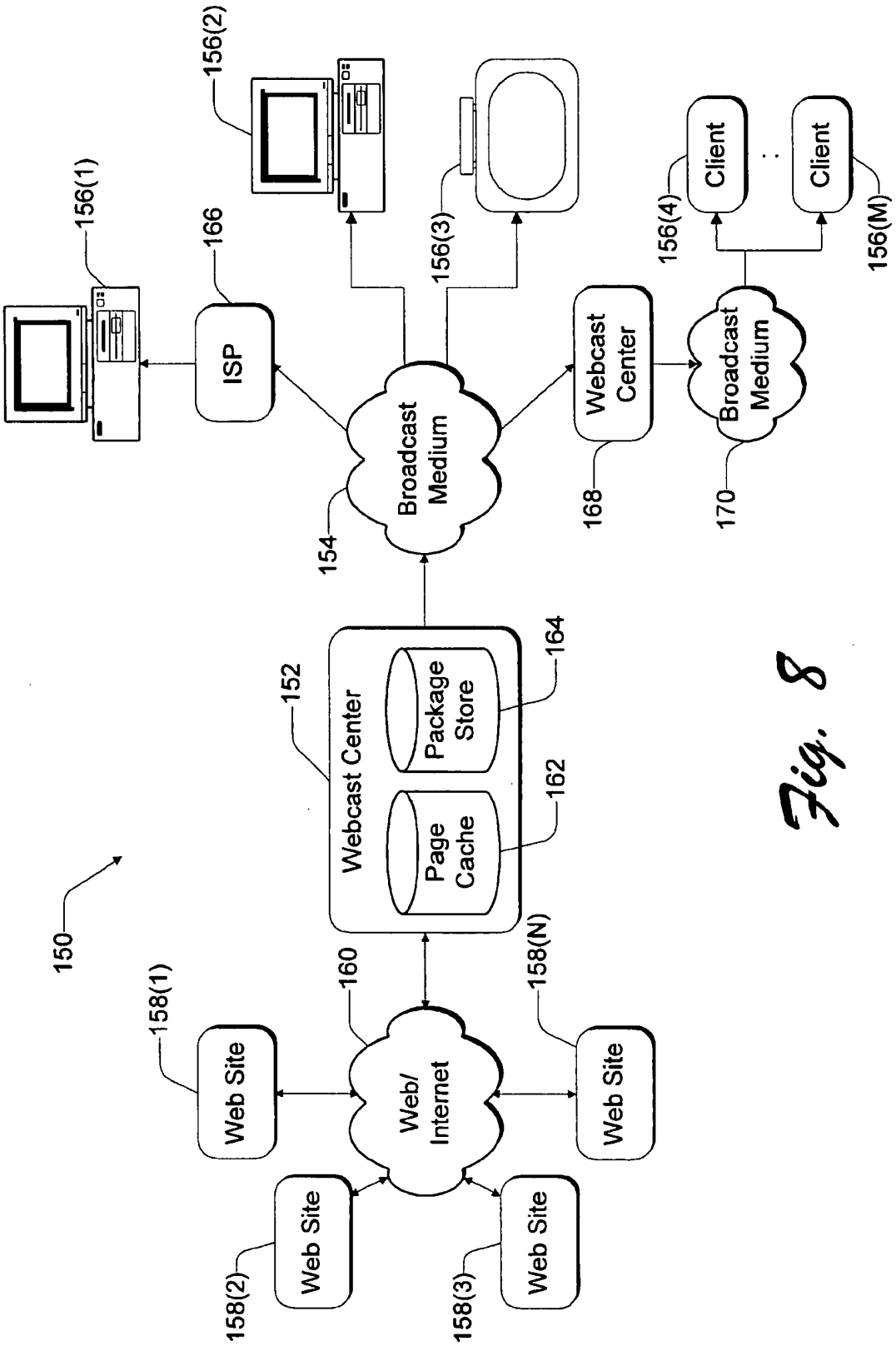

150
Webcast Center
152
Package Store
164
Page Cache
162
Broadcast Medium
154
ISP
166
Webcast Center
168
Broadcast Medium
170
Web/ Internet
160
Web Site
158(1)
Web Site
158(2)
Web Site
158(3)
Web Site
158(N)
156(1)
156(2)
156(3)
156(4)
Client
Client
156(M)

Fig. 8

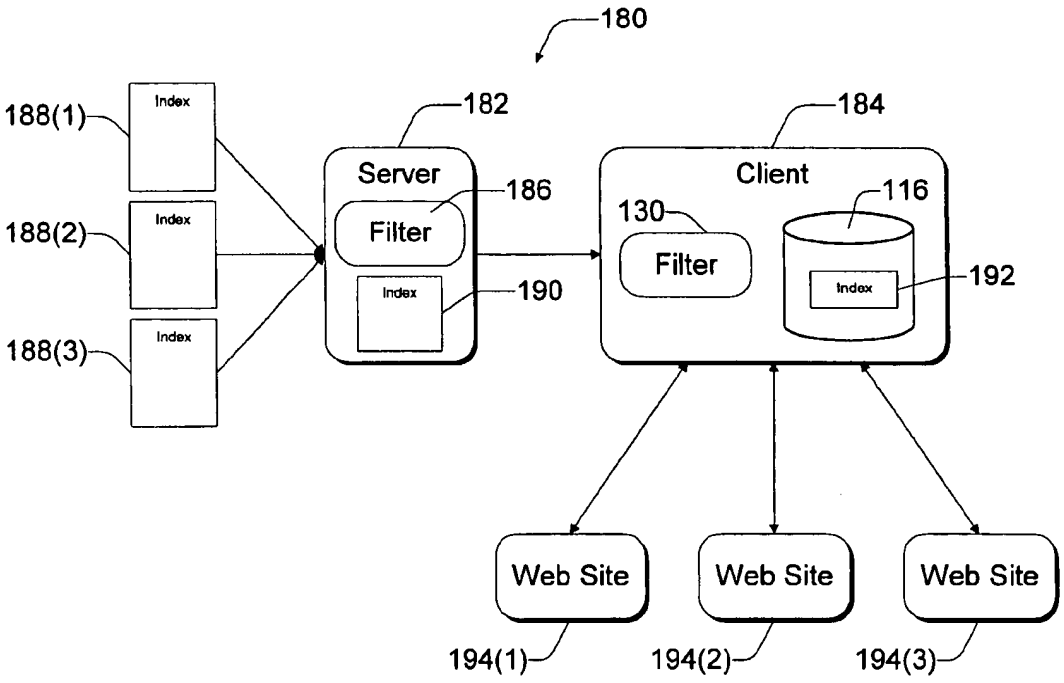

180
188(1)
Index
188(2)
Index
188(3)
Index
182
Server
Filter
Index
186
190
184
Client
130
Filter
116
Index
192
Web Site
Web Site
Web Site
194(1)
194(2)
194(3)

Fig. 9

CLIENT-SIDE SYSTEM FOR SCHEDULING DELIVERY OF WEB CONTENT AND LOCALLY MANAGING THE WEB CONTENT

TECHNICAL FIELD

This invention relates to client-server systems and methods for obtaining Web related content from one or more servers and presenting that content to a user. More particularly, this invention further relates to client-side software and devices that facilitate delivery and presentation of the Web content.

BACKGROUND OF THE INVENTION

Public networks, and most notably the Internet, are emerging as a primary conduit for communications, entertainment, and business services. The Internet is a network formed by the cooperative interconnection of computing networks, including local and wide area networks. It interconnects computers from around the world with existing and even incompatible technologies by employing common protocols that smoothly integrate the individual and diverse components.

The Internet has recently been popularized by the overwhelming and rapid success of the World Wide Web (WWW or Web). The Web links together various topics in a complex, non-sequential web of associations which permit a user to browse from one topic to another, regardless of the presented order of topics. The Web is rapidly evolving as a standard for distributing, finding, and accessing information of any type. A "Web browser" is an application that executes on the user's computer to navigate the Web. The Web browser allows a user to retrieve and render hypermedia content from the WWW, including text, sound, images, video, and other data.

The amazing growth rate in the demand for data over the Internet is partly due to an increasing audience. The World Wide Web has crossed the threshold that makes it affordable and interesting to a much larger audience. There is information available on a very wide variety of topics, and tools exist to help people find and view the information cost effectively.

Another factor fueling the Internet growth is the exploding amount of information that is now available on the Web. The Web has grown from thousands of Web sites to several million Web sites in a very short period of time. The growth continues at an exponential rate. Many corporations and libraries are translating paper and microfilm information archives to electronic media that is published via the Web or similar network. While this has resulted in a wealth of information that is now available to virtually anyone, the information is poorly organized and the sheer volume of the information makes it hard for a typical person to sort through, find, and retrieve specific information.

The shift from paper published media to online media also created a new problem. People wishing to access Web information are limited to accessing it only when connected to the Internet or other network. Network connectivity is largely restricted to a physical wire connection to the computer, or a virtual connection to wireless transmission networks. This makes it hard, if not impossible, to disconnect the computer from the network and still access information.

As more information is brought online, the demand on the computational and network resources to categorize, search, personalize, and retrieve the information is placing new demands on the existing client-server infrastructure that makes up networks like the Web. Additionally, the data demands are affected by a trend for Web sites to evolve from serving pure text to serving richer media content, including graphics, sound, and video. Adding richer media content is popular because it presents information more clearly and attractively, thereby enhancing a site's impact and popularity.

Due to these emerging factors, a significant problem facing the continued growth and acceptance of the Internet is that conventional methods for accessing the Web do not scale well to meet the rapid growth in supply and demand, or to satisfy the need for better organization. The quality of service for the Web is intuitively measured by the user as the amount of time it takes to search, find, request, and receive data from the Web. Internet users have been conditioned through their experiences with television and standalone multimedia applications to expect instantaneous results on demand. Users are accustomed to changing the TV channel and instantaneously viewing the video content for that channel on the screen. Unfortunately, the Internet is unable to deliver data instantaneously. For the most part, the Internet has significant latency problems that reduce fairly routine Web browsing exercises to protracted lessons in patience.

The basic dilemma is that the quality of service degrades as more people try to use the Web. More unsettling is the corollary that service for popular Web sites is typically much worse than service for unpopular sites. There are several causes of the service problem, including overburdened servers and slow distribution networks.

Networks often have too little bandwidth to adequately distribute the data. "Bandwidth" is the amount of data that can be moved through a particular network segment at any one time. The Internet is a conglomerate of different technologies with different associated bandwidths. Distribution over the Internet is usually constrained by the segment with the lowest available bandwidth.

In the consumer market, for example, most clients typically connect to the Internet via a local modem connection to an Internet Service Provider (ISP). This connection is generally enable a maximum data rate of 14.4 Kbps (Kilobits per second) to 28.8 Kbps. Some clients might employ an ISDN connection, which facilitates data flow in the range of 128–132 Kbps.

The ISP connects to the primary distribution network using a higher bandwidth pipeline, such as a T1 connection that can facilitate a maximum data flow of approximately 1.5 Mbps. This bandwidth is available to serve all of the clients of the ISP so that each client can consume a 14.4 Kbps, 28.8 Kbps, or 128 Kbps slice of the 1.5 Mbps bandwidth. As more clients utilize the ISP services, however, there is less available bandwidth to satisfy the subscriber requests. If too many requests are received, the ISP becomes overburdened and is not able to adequately service the requests in a timely manner, causing frustration to the users.

Couple this problem with the fact that clients typically go underutilized. While servers are pushed to their maximum output limits, clients often sit idle for many hours per day.

Because the bandwidth issue is constrained by technology development in the physical network architecture, early attempts to solve these problems focused on organizing the Web content in some manner to better facilitate search and retrieval. This in turn enabled users to more quickly access information on the Internet, even though the underlying physical architecture remained the same.

The earliest solutions involve organizing the information by hand. Humans review information by browsing the Internet and assemble large lists of documents containing similar information. The lists are further organized into hierarchies of categorized content. People can view the categorized lists online in an attempt to more quickly obtain a specific piece of information. The advantage of this scheme is that human reviewers are very good at categorizing the information and discarding low-value documents, so the lists of categorized information contain fairly high value information. Some hand-categorized data schemes are organized into popular Web sites. The best known example of this is the "Yahoo!" Web site.

The disadvantage of this human-driven technique is that it becomes more difficult to keep up when the amount of information grows exponentially. The categorized lists are frequently out of date or inadequate. Additionally, the method requires a user to be connected to the network to view the information.

Another approach is to use massive search engines that automatically retrieve documents on the Web and attempt to index all of the information. The technique of fetching this information is known as "web-crawling" or "web-scraping". Heuristic document categorization algorithms index the information and store the indices (but not the information) in large centralized databases. Users run queries against the massive databases to find specific information, and then retrieve the information from individual web-sites. Popular examples of these types of Web based services include Lycos, InfoSeek, Alta-Vista, and others. They are generally referred to as "Search Sites" or "Internet Search Engines".

The advantage of web-crawling and indexing is that computers can automate the process of retrieving and reviewing documents. The speed of computers means that a larger number of documents can be compiled as compared to human efforts. The disadvantage is that the computers have a hard time distinguishing between valuable information and worthless information, and are not very good at categorizing the information. Also, these types of databases are centralized and require an end user to be online to make queries against the database. A third approach to solving the information glut problem is to employ information services that collect and editorialize information that they deem as important. The information is indexed and placed into a centralized database. The services utilize a combination of humans to collect and categorize information, and computers to perform automated information collection. Because these systems effectively filter down the amount of potential information by many orders of magnitude, it is possible to locally store portions of the centralized database on the client server and for the user to view the information when disconnected.

The most popular example of this type of system is PointCast. PointCast collects news articles from many sources, edits them down to a predefined maximum length, categorizes them, and stores them in a centralized database at their data center. Client software then queries the centralized database to obtain the portions of the data in which the user is interested.

The disadvantage of these systems is that a centralized database scales poorly as more and more users attempt to retrieve information. By centralizing all information, the data source becomes a choker point to information flow. Another disadvantage is that while some of these centralized information services provide a good selection of information for users, the information is dramatically more restricted in comparison to the vast wealth of information available on the Web. Users are restricted to these service-selected information categories.

Accordingly, there remains a need to develop improved techniques for facilitating distribution of Web content over the Internet.

SUMMARY OF THE INVENTION

This invention concerns a client-based system that improves gathering and organizing of Web content in a manner that mitigates impact on overburdened servers and slow networks. The client-based system enables personalized filtering to collect only that content which the individual user prefers, while rejecting unwanted content. Moreover, the system enables the user to work offline from the server with similar functionality to online operation.

According to one aspect of this invention, the client-based system has a scheduling subsystem to schedule a time to obtain the Web content from the server. When the client reaches the scheduled time, the scheduling subsystem generates an event notification that contains sufficient information explaining how to retrieve the Web content. As an example, the event notification might contain a URL (universal resource locator) that the client uses to go out and fetch the Web content. The event notification might alternatively contain a reference to a multicast address or a broadcast transmission frequency to which the client listens or tunes to retrieve the desired Web content.

The client-based system has a delivery subsystem that is responsive to the event notification to facilitate retrieval of the Web content at the time set by the scheduling subsystem. The delivery subsystem preferably has multiple delivery modules that enable delivery of the content over different types of distribution systems. For instance, the delivery subsystem might comprise a multicast listener to listen to a multicast address for the Web content, or a fetching program that goes out to the server and retrieves the Web content over the Internet, or a broadcast packet rebuilder that reconstructs Web content that is broadcast over a wireless network.

In addition to the Web content or data itself, the delivery subsystem obtains an index to the Web content. The index summarizes the Web content to facilitate local search and find tasks. The index and Web content are stored in a cache at the client, preferably according to some unique identifier such as URLs.

The client-based system also has an indexing subsystem to retrieve the index from the cache and present the index to a user. The indexing subsystem supports a user interface, such as a graphical windowing UI, which enables the user to select from the index portions of the Web content stored in the cache.

According to an aspect of this invention, the user can create personal filters that filter the index to remove items not of interest. The filters can condense the index when it is received prior to be cached, or when the user attempts to view the index.

According to another aspect of this invention, the user can continue to search and find the Web content using the index even though the client is offline from the server. The user is given essentially the same functionality as a live online session, except that requests to remote servers are temporarily accumulated for later submission. For example, the user may fill out an HTML (hypertext markup language) form and click a "submit" button to send the completed form back to the originating Web site. To the user, the clicking action appears to send the form back to the server. However,

since the client is offline, the HTML form is kept in the cache until a later online session. When the client subsequently reconnects to the server, all accumulated data (i.e., requests, forms, etc.) that is destined for one or more remote servers is sent in batch to the appropriate servers.

According to another aspect, the user can create his/her own channel. The client-based system enables the user to select preferred Web content that is delivered using different channels. For instance, the user might like to see all basketball-related content. Based on the user's selections, the system constructs a set of filtration rules and filters the different channels according to the filtration rules to aggregate the preferred Web content. In this manner, the system might extract basketball scores from one Web site, player statistics from another, and upcoming schedules from a third. The client-based system then presents the aggregated Web content as a new channel to a user, such as the "Basketball" channel.

In one implementation, the client-based system is built into a Web browser. The browser may be integrated into the operating system, or run as a separate application.

BRIEF DESCRIPTION OF THE DRAWINGS

The same reference numbers are used throughout the drawings to reference like components and features.

FIG. **1** is a diagrammatic illustration of a client-server system.

FIG. **2** is a block diagram of a client computer.

FIG. **3** is a block diagram of a client-based system for obtaining and caching Web content. FIG. **3** shows the client-based system implemented in a browser.

FIG. **4** is a diagrammatic illustration of a graphical user interface used to schedule when to obtain Web content.

FIG. **5** is a diagrammatic illustration of a graphical user interface used to present an index of the Web content to a user.

FIG. **6** is a diagrammatic illustration of a graphical user interface used to present the Web content to the user.

FIG. **7** is a flow diagram in a client-side process for subscribing to Web content, scheduling its delivery, and presenting it to the user.

FIG. **8** is a diagrammatic illustration of a webcast system.

FIG. **9** is a diagrammatic illustration of a client-server system in which the server implements filters constructed according to client preferences.

DETAILED DESCRIPTION

FIG. **1** shows a client-server system **20** having multiple Web servers **22**(1)–**22**(M) coupled to serve Web content to multiple clients **24**(1)–**24**(N) via a distribution system **26**. The Web content can come in many different forms. One example is a Web page stored at a Web site. A Web page is a title, collection of information, and pointers or "hyperlinks" to other information. A Web page may be constructed from various types of content including computer data, audio, video, animation, bit maps or other graphics, applications or other executable code, text, hypermedia, or other multimedia types. Another example of Web content is a video or audio that can be played at the server and transmitted over a distribution system **26** to one or more clients.

Distribution system **26** represents many different types of distribution systems. As an example, the distribution system **26** might represent the Internet, or an Intranet, or other network. Such networks enable point-to-point communication, one-to-many communication, and many-to-many communication. The Internet, for example, supports multicast transmissions in which one or more servers transmit content to a predefined address. Clients listen to the address to receive the multicast content. In addition, such network systems (excepting perhaps multicast) are typically characterized as bi-directional, allowing communication both from the server to the client, and return communication from the client back to the server.

The distribution system **26** might also represent a broadcast transmission system in which Web content is distributed over a broadcast medium, such as radio, TV, microwave, satellite, or the like. A broadcast distribution system supports one-to-many communication and is generally characterized as a unidirectional system. Multicast is usually likened to a broadcast system as being unidirectional.

According to an aspect of this invention, the Web servers provide both the Web content **28** and an index **30** to the Web content. The index **30** contains information about the Web content **28**. The index **30** also provides a way to locate the actual Web content, such as specifying a URL or a channel for each piece of Web content that is listed. The index **30** includes descriptive information about each item of content, such as title, author, summary, last time modified, etc. This descriptive information can be used to categorize the Web content.

The client-server system **20** supports a two-phase delivery, regardless of which type of distribution system is employed. The first phase is to deliver the index **30**. The index may originate from one server, or it may be a collection of elements originating from multiple servers. The index can then be used to identify the Web content **28** to be delivered to the client. The second phase is to deliver the Web content **28**. The Web content may originate from one server, or from multiple servers. Moreover, the index and Web content may originate from the same server or from separate servers.

The distribution system **26** supports different transfer architectures. The delivery of the index **30** and the Web content **28** can involve one or more of the following architectures: a "pull-based" architecture, a "poll-based" architecture, and a "push-based" architecture. In a pull-based architecture, the user directly or indirectly instructs the client software to initiate a request for data from the server. HTTP (hyptertext transfer protocol) and FTP (file transfer protocol) are examples of a "pull-based" architecture.

In a poll-based architecture, the client software "pulls" the data on a periodic basis, not directly initiated by a user action. This may be based on a fixed repeating schedule, or a repeating schedule with a random element. Polling HTTP is an example of a "poll-based" architecture.

In a push-based architecture, the server initiates data transfer to the client software. Multicast protocols, wireless pagers, radio, and TV are examples of "push-based" architecture. To the casual user, "poll" and "push" can be made to appear the same.

The client-server system **20** employs a channel metaphor to generally describe how the Web content **28** and index **30** are made available to the user. For instance, news-related Web content might be available on a news channel and sports content might be available on the sports channel. In some instances, the channel is associated with a particular source, such as a CNN channel that facilitates delivery of CNN news from the CNN Web site. However, the term "channel" is not restricted to a single source, or to a single transport mechanism, or to a single protocol.

More broadly-speaking, a "channel" is an organizational tool that defines how content is bundled for presentation to the user. From the user perspective, the channel defines a content class, even though the content may be the aggregation of data from many different sources.

As possible examples, a channel might represent the content that is available from a single Web site, such as a channel for the popular Web site "ESPN SportsZone". The channel might alternatively consist of a group of like content that the user personally assembles and which is gathered from multiple sources. For instance, the user might create a "Basketball" channel that collects and presents basketball-related content from various sources like ESPN, CNN, MSNBC, and the like.

The channel might further represent a physical transport, such as a channel associated with a multicast address or a channel associated with a particular airwave frequency. In this regard, the term channel is akin to the familiar TV-notion of channel. But, the term "channel" is not restricted nor necessarily tied to the underlying transport mechanism and hence is more general than the traditional TV channel.

Exemplary Client Configuration

FIG. **2** shows an example implementation of the client computer, referenced generally as number **24**. The client is illustrated as being implemented as a general-purpose computer. The client **24** includes a processing unit **32**, a system memory **34**, and a system bus **36** that interconnects various system components, including the system memory **34** to the processing unit **32**. The system bus **36** may be implemented as any one of several bus structures and using any of a variety of bus architectures, including a memory bus or memory controller, a peripheral bus, and a local bus.

The system memory **34** includes read only memory (ROM) **38** and random access memory (RAM) **40**. A basic input/output system **42** (BIOS) is stored in ROM **38**.

The client **24** has one or more of the following drives: a hard disk drive **44** for reading from and writing to a hard disk or hard disk array, a magnetic disk drive **46** for reading from or writing to a removable magnetic disk **48**, and an optical disk drive **50** for reading from or writing to a removable optical disk **52** such as a CD ROM or other optical media. The hard disk drive **44**, magnetic disk drive **46**, and optical disk drive **50** are connected to the system bus **36** by a hard disk drive interface **54**, a magnetic disk drive interface **56**, and an optical drive interface **58**, respectively. The drives and their associated computer-readable media provide non-volatile storage of computer readable instructions, data structures, program modules and other data for the client **24**.

Although a hard disk, a removable magnetic disk **48**, and a removable optical disk **52** are described, other types of computer readable media can be used to store data. Other such media include magnetic cassettes, flash memory cards, digital video disks, Bernoulli cartridges, random access memories (RAMs), read only memories (ROM), and the like.

A number of program modules may be stored on the hard disk, magnetic disk **48**, optical disk **52**, ROM **38**, or RAM **40**. These programs include a server operating system **60**, one or more application programs **62**, other program modules **64**, and program data **66**. The operating system **60** is preferably a multitasking operating system that allows simultaneous execution of multiple application programs **62**. The operating system employs a graphical user interface windowing environment that presents the applications or documents in specially delineated areas of the display screen called "windows." One preferred operating system is a Windows brand operating system sold by Microsoft Corporation, such as Windows 95, Windows CE, Windows NT or other derivative versions of Windows. It is noted, however, that other operating systems may be employed.

A user may enter commands and information into the server **22** through input devices such as a keyboard **68** and a mouse **70**. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are connected to the processing unit **32** through a serial port interface **72** that is coupled to the system bus **36**, but may alternatively be connected by other interfaces, such as a parallel port, game port, or a universal serial bus (USB).

A monitor **74** or other type of display device is also connected to the system bus **36** via an interface, such as a video adapter **76**. In addition to the monitor, personal computers typically include other peripheral output devices (not shown) such as speakers and printers.

The client computer **24** has a network interface or adapter **78**, a modem **80**, or other means for establishing communications over a network **82** (e.g., LAN, Internet, etc.). The modem **80**, which may be internal or external, is connected to the system bus **36** via the serial port interface **72**.

Although not shown, the client **24** may also be implemented as a broadcast-enabled computer, which includes a digital broadcast receiver (e.g., satellite dish receiver, RF receiver, microwave receiver, etc.) and a tuner which tunes to appropriate frequencies of the broadcast network. One example implementation of a broadcast-enabled PC is described in a co-pending U.S. patent application Ser. No. 08/653,663, filed Jan. 29, 1996, which is a continuation of U.S. patent application Ser. No. 08/503,055, entitled "Broadcast-Enabled Personal Computer," filed Jul. 17, 1995, which is now abandoned. These applications were filed in the names of Gabe L. Newell, Dan Newell, Steven J. Fluegel, David S. Byrne, Whitney McCleary, James O. Robarts, Brian K. Moran; William B. McCormick, T. K. Backman, Kenneth J. Birdwell, Joseph S. Robinson, Alonzo Gariepy, Marc W. Whitman, and Larry Brader. This application is assigned to Microsoft Corporation, and is incorporated herein by reference.

Client-Based System

An aspect of this invention concerns a client-based system, implemented at each of the clients **24(1)–24(N)**, which improves gathering and organizing of the Web content **28**. For purposes of continuing discussion, the client-based system is described in the context of being incorporated into a Web browser, such as the Internet Explorer browser available from Microsoft Corporation. FIG. **2** shows a Web browser **90** implemented as a separate application **62** or integrated into an operating system **60**. However, it is noted that aspects of this invention can be implemented apart from a Web browser.

FIG. **3** shows the Web browser **90** in more detail. It includes a scheduling subsystem **92** to schedule a time to gather the Web content from one or more servers. It also includes a delivery subsystem **94**, which is responsive to the scheduling subsystem **92**, to obtain the Web content at the scheduled time.

The scheduling subsystem **92** has a scheduler module **96** and a notification generator module **98**. The scheduler **96** consists of software code that manages when the delivery subsystem **94** is to run at a later time. The scheduler **96** thus sets the time event when certain Web content is to be collected. This may be a one-time event, a periodic event, or even an event whose occurrence is based on some degree of randomness.

The scheduler **96** supports a graphical user interface (UI) that enables a user to schedule such time events. FIG. **4** shows an example of a scheduling UI **100** that allows the user to specify when the browser should collect content from the Internet. The scheduling UI **100** has a field **102** that permits the user to define and name different schedules. The UI **100** also has multiple parameters **104** that the user can elect to establish various collection times.

In some cases, the user may wish to schedule the gathering of Web content at predictably low traffic times, such as at midnight or early morning hours. The user enters these constraints in the "Time" field of the schedule UI **100**, as shown. The ability to coordinate delivery of content at off-hours helps alleviate network congestion and the burden on servers.

With reference again to FIG. **3**, when the scheduled time arrives, the scheduler **96** informs the notification generator **98** to generate an event notification. The event notification contains sufficient information to configure, or obtain configuration information, for the delivery subsystem **94** to begin retrieval of the index and content. The event notification might contain one or more of the following types of information:

a channel reference

instructions telling the delivery subsystem which mechanism to use to obtain the data (e.g., fetching, broadcast, multicast)

one or more URLs

a multicast address;

a wireless frequency (radio, TV, etc.)

The delivery subsystem **94** provides the means for obtaining the index and Web content. The delivery subsystem **94** supports one or more different mechanisms to retrieve the information. In the illustrated implementation, the delivery subsystem **94** includes a Web fetching program **110**, a broadcast packet rebuilder **112**, and a multicast listener **114**.

The Web fetching program **110** enables the basic functionality of going out on the Web and getting the desired content. The Web fetching program **110** uses URLs to locate the index and Web content, and downloads the found information.

The broadcast packet rebuilder **112** is used to reassemble Web content from packets that are broadcast over a broadcast medium. In the case where data is bundled and broadcast over a broadcast medium (e.g., radio, microwave, TV, etc.), the client is equipped with a broadcast receiver to receive the packets. The broadcast receiver routes the packets to the packet rebuilder **112**, which reconstructs the data from the packets.

The multicast listener **114** is a program that tunes to designated multicast addresses on the network to receive messages.

When the delivery subsystem **94** retrieves the index **30** and Web content **28**, it stores them in a local cache **116**. The cache **116** is implemented in the hard disk drive **44** of the client computer **24**, to provide persistent storage of the data. It is noted, however, that other storage means may be used to implement the cache **116**, such as RAM **40** and magnetic disk drive **46**.

The delivery subsystem **94** stores the Web content **28** according to a corresponding unique identifier. As one example, the Web content **28** is stored according to URLs. In this manner, the client browser can access locally cached copies of the Web content using the same URLs that would be used to retrieve the same content from remote servers.

The browser **90** also has a content indexing subsystem **120** to retrieve the index from the cache **116** and present the index to a user through a user interface **122**. The index lists the available Web content that is stored in the cache, and enables the user to select or reject certain types of content.

FIG. **5** shows an example of an index viewer UI **122**, which presents the Web content in a hierarchical organization. In this example, the index viewer UI **122** is a "pane" of a larger graphical user interface window, as is shown more clearly in FIG. **6**.

The index UI **122** presents general categories, such as "News and Technology", "Sports", "Business", "Entertainment", "Lifestyle and Travel", "The Microsoft Network", and "MSNBC". There is also a category that contains a "Channel Guide", which provides information on the various channels available to the user. The user can elect certain channels and content by appropriately marking them in the index viewer UI **122**.

The indexing subsystem **120** stores the user's preferences in a preference store **124** (which may be physically implemented in the cache **116** or other memory of the client computer). The browser **90** uses the user preferences to collect any additional Web content that is not locally stored in the cache **116**. Additionally, the preferences are used to create filters that remove unwanted Web content before it is presented to the user.

The browser **90** has a filtering subsystem **130** that creates and maintains one or more personalized filters **132** and **134**. The filtering subsystem **130** collects the user's preferences from the preference store **124** and constructs filters **132** and **134** based on the preferences. The filters scan the index **30** or Web content **28** and identify matches between the user's preferences and information stored in the index **30** or Web content **28**. Index items or content data that do not match the user's preferences are discarded.

One type of filter is a "pre-cache" filter that filters incoming information as it is received from servers and prior to storage on the cache **116**. Filter **132** is an example of a pre-cache filter. With the incoming filter **132**, unwanted index items or Web content is rejected before it is stored locally.

Another type of filter is a "post-cache" filter that filters the index **30** and Web content **28** stored on the cache **116** prior to presenting it to the user. Filter **134** is an example of a post-cache filter.

The filtering subsystem **130** can be configured to filter on language types. For instance, the user might choose to view only content presented in a particular language, such as English or Spanish. Some Web sites contain multi-language documents and links to other multi-language data. With the language filter activated, any Web content in a language other than the selected language is rejected.

The browser **90** also has a content viewer UI **140** that presents the Web content to the user. The content viewer UI **140** is preferably the same windowing UI employed during normal browser operation.

FIG. **6** shows an example of the content viewer UI **140**, which presents the Web content to the user. In the example of FIG. **6**, the content viewer UI **140** is embodied in the Internet Explorer browser, with the familiar menu, toolbar, and task bar.

The viewer UI **140** includes a presentation space **142** that depicts the Web content. In this example, the content is from a Disney channel, as indicated by the channel pane **122** adjacent the content space **142**.

Exemplary Scenario

FIG. **7** shows an example process enabled by the client-based system described above. At step **200**, a user indicates, directly through a user interface or indirectly as a byproduct

of some other action, that he/she wants to subscribe to some type of Web content. The subscription process involves downloading information, typically in the form of HTML forms, from the host Web site and invoking a Registration Wizard to step the user through the subscription forms. The user enters the requested information and the completed forms are sent back to the Web site.

The host site provides a schedule for its Web content. If the content is to be broadcast or multicast, the schedule indicates the times and the frequency or address at which the Web content will be made available. The schedule from the host site is stored as part of the index **30** in the cache.

At step **202**, the scheduling subsystem **92** schedules retrieval of desired Web content at certain times. The times might be those specified by the user (e.g., off-hour retrieval times) or those specified as the broadcast or multicast times. The scheduler **96** then tracks when the schedule times arrive (step **204**).

When a schedule time arrives (i.e., the "yes" branch from step **204**), the notification generator **98** generates a notification event (step **206**). This notification event is passed to the delivery subsystem **94**, which invokes the appropriate delivery module to begin the process **208** of obtaining the information.

The delivery process **208** involves two phases. The first phase is to retrieve the index **30** (step **210**). The second phase is to retrieve the Web content **28** (step **212**). The browser stores the index and Web content in the cache **116** (step **214**).

The filtering subsystem **130** may be invoked to filter the index and/or content at different phases. One or more filters might be applied to the index prior to determining what content to pull from the Internet (step **216**(*a*)). In addition, one or more filters might be applied after both the index and Web content are retrieved, but prior to caching (step **216**(*b*)). As a third alternative, one or more filters might be applied to the index and/or content after caching but prior to presentation to the user (step **216**(*c*)).

At step **218**, the index is retrieved from the cache and presented to the user in the index viewer UI **122**. The index viewer UI **122** displays one or more indices that are associated with the information to which the user has subscribed. Once the user has found some information they deem valuable, the user selects the Web content (i.e., the "yes" branch from step **220**). The selected Web content is then presented to the user in the content viewer UI **140** (step **222**).

Aggregation/Disaggregation

The browser **90** enables the user to construct custom or personal channels by aggregating content from multiple channels into a single custom channel. The user selects a set of channels from the channel pane **122** and indicates the preferred Web content within each channel. The browser takes the user's input and constructs a set of filtration rules based on the user's selections and preferences. The browser then creates a new channel that presents the Web content from the set of channels that survives the filters.

As an example, suppose the user wants a personal channel that contains only basketball-related content. The user selects a set of channels that might carry basketball information, such as ESPN, CBS, CNN, and the like. Within each channel, the user can mark the sub-channel for basketball content or apply a filter for specific items in that channel to be disaggregated and then reaggregated. In FIG. **5**, for instance, the user might check CBS SporstLine Channel, and the sub-channels "NBA" and "College Basketball". In the case of the filter, basketball-related content is automatically identified by the browser based on keywords, tags, or other means for identification that the content provider might include with the content. These preferences are stored in the preference store **124**.

The filtering subsystem **130** creates one or more filters that identify the basketball information from each of the selected channels. The new channel then references the identified basketball information by maintaining, for example, the URL to the basketball information as it is stored in the cache **116**.

The channel pane UI **122** lists the personal channel as the "Basketball" channel. It may also identify sub-channels such as EPSN highlights, CBS Game of the Week, and so forth. When the user clicks on the Basketball channel or sub-channel, the browser retrieves the basketball content and presents it in the viewer UI **140**.

In addition to aggregating content from several channels into a custom channel, the browser **90** allows the user to disaggregate content from a single channel. Disaggregation might be used to change the offerings of a channel, or to modify the channels' hierarchical categorization of content, or to create multiple channels from a single channel. This all occurs at the client, so the server-side organization is not altered.

As an example of disaggregation, suppose a channel for offers news and sports as a sub-channel to the news. The user can choose to delete the news channel, while preserving the sports channel. Alternatively, the user might move the sports channel to a different level, such as equal to the news so that it is no longer a sub-channel to the news. The user might further choose to disaggregate the news and sports into two separate channels.

Offline Submission

The browser **90** allows a user to work offline from the server in a manner that feels familiar to working online. After the Web content **28** is downloaded and stored in the cache **116**, the client can disconnect from the server or network. Despite being disconnected, the user can continue to search and find the Web content using the locally cached data. The Web content can be, for example, in the form of Web pages with internal hyperlinks to other pages in the cache. Accordingly, the user can browse through the Web content in the cache **116**, while offline, in the same manner that he/she browses the content while online.

When the user performs operations that involve submitting data to a remote server, the browser temporarily accumulates the outgoing data **146** in the cache **116** for submission at a later time. For example, during the course of browsing, the user may stumble onto a service that he/she would like to join. The user fills out the form, such as an HTML form, and clicks a "submit" button to send the completed form back to the originating Web site. To the user, the clicking action appears to send the form back to the server, as the form leaves the screen as if it were sent.

Since the client is offline, the HTML form is not really sent to the server. Instead, it is kept in the cache **116** until a later online session. When the client subsequently reconnects to the network during the next online session, all of the accumulated data **146** that is destined for one or more remote servers (i.e., requests, forms, etc.) are sent in a batch to the appropriate servers.

Webcast Center Implementation

The client-based system described above is also well suited for use in a webcast system. FIG. **8** shows a webcast system **150** for delivering Web content from a webcast center **152** over a broadcast medium **154** to multiple clients **156**(1)–**156**(M). The webcast center **152** gathers Web content from the World Wide Web by visiting web sites **158**(1)

–158(N) via the Internet **160** and fetching content from those sites. The webcast center **152** collects Web pages from the Internet's World Wide Web **160** and stores them in a page cache **162**. A system administrator sets a schedule that establishes which sites are visited by the webcast center **152**, the time and frequency of the visits, and the type of content collected.

Apart from the gathering process, the webcast center **152** retrieves the pages from the page cache **162**, bundles them into composite package files, and stores them in a package store **164**. The package store **164** is preferably a separate database than the page cache **162**. The webcast center **152** fetches the package files from the package store **164**, segments the package files into individual packages (or packets), and transmits the packages over the broadcast medium **154**.

The broadcast medium **154** is a unidirectional network in which packages are delivered from the webcast center **152** to the clients **156(1)–156(M)** without requiring return communication from the clients. The broadcast medium **154** can be characterized as a shared, highly asymmetrical, network resource with a limited, if not completely absent, low speed return path that does not need to be active to receive broadcast transmissions. The broadcast medium **154** may comprise the entire distribution network between the webcast center and clients, or it may be a single link in a larger distribution network.

The broadcast medium **154** may be implemented in a variety of ways. The broadcast medium **154** might be implemented, for example, as a wireless network configured for one-way transmission (i.e., satellite, radio, microwave, etc.). The broadcast medium **154** might also be configured as a network that supports two-way communication (i.e., Internet, LAN (local area network), and WAN (wide area network)), but can be used for unidirectional multicasting from the webcast center to the clients.

The clients **156(1)–156(M)** represent various types of constructions. The clients can be implemented as essentially any type of computing device that can receive and reconstruct data packages, and render the packages on a display. As one possible implementation, the client may be constructed as a desktop computer, as represented clients **156(1)** and **156(2)**, that are specially configured with software/hardware components described below with respect to FIG. **2**. Client **156(1)** receives broadcast Web content from the broadcast medium **154** via an Independent Service Provider (ISP) **166**, rather than receiving the broadcasts directly. On the other hand, client **156(2)** is a broadcast-enabled personal computer that is capable of receiving the broadcast packets directly.

Another implementation of a client is a Web-enabled television, as represented by client **156(3)**, which has a set-top box or internal computing unit that permits receipt and rendering of Web content. In addition to desktop computers and Web-enabled TVs, other possible clients include workstations, laptop computers, palmtop computers, network computers, and the like.

Another distribution entity may act as a "client" to the webcast center **152**. As shown in FIG. **8**, the regional Independent Service Provider (ISP) **166** might be a subscriber to the broadcast transmissions received over the broadcast medium **154** from the webcast center **152**. The ISP **166** stores the webcast content and distributes it to its own clientele, such as client **156(1)**, using conventional distribution techniques.

As another example of an intermediary distribution entity, a secondary webcast center **168** may function as a "client" to the primary webcast center **152**. In addition to its own independent gathering process, the secondary webcast center **168** also receives and re-broadcasts the Web content received from the primary webcast center **152** to a set of clients **156(4)–156(M)** over a broadcast medium **170**. One implementation of this dual webcast center architecture is that the primary webcast center **152** is a primary head end that distributes nationally or globally via satellites, and the secondary webcast center **168** is a regional distributor that distributes the Web content via RF (radio frequency) or microwave transmission.

A more detailed discussion of this webcast system **150** is provided in a co-pending U.S. patent application Ser. No. 08/958,609, entitled "System and Method for Delivering Web Content over a Broadcast Medium", which was filed Oct. 27, 1997, in the names of Anne Wright, Randy Sargent, Carl Witty, Brian Moran, and David Feinleib. This co-pending application is assigned to Microsoft Corporation and is incorporated by reference.

Server-Side Filtering Based on Client Preferences

As discussed above, the browser **90** enables the user to define certain preference criteria that is used to create filters. In the above implementation, the filters **132, 134** reside at the client. In another implementation, these user preferences can be used to create filters on the server side.

FIG. **9** shows a client-server system **180** having a server **182** and a client **184**. The client **184** is constructed as described above, having both a cache **116** and a local filtering subsystem **130**. The client **184** establishes an account or some form of registration with the server **182**. The client **184** then submits the user's preferences to the server **182**, which creates one or more filters **186** based on the user's preferences. These filters **186** are maintained at the server **182** under the client's account.

As the server receives various indexes **188(1)–188(3)** of available Web content, the server **182** filters the indexes using the server-side filters **186** to create a customized index **190**. The server **182** occasionally downloads the customized index **190** to the client **184**.

At that point, the client **184** may additionally apply its local filters **130** to further condense the customized index to yet a smaller index **192**. It is this doubly-filtered index **192** that is presented to the user. Depending on the user's selection, the client obtains the Web content either from the local cache, if available, or directly from the Web sites **194(1)–194(3)** themselves. Notice that the server supplying the filtered index need not be the actual Web sites that hold the information, although it can be. For instance, the client can use the condensed index **192** as a means for identifying the Web content to be pulled down to the client for the user's perusal. Once the Web content is identified, the client schedules retrieval of the content from one or more Web sites **182** and **194(1)–194(3)**.

Although the invention has been described in language specific to structural features and/or methodological steps, it is to be understood that the invention defined in the appended claims is not necessarily limited to the specific features or steps described. Rather, the specific features and steps are disclosed as preferred forms of implementing the claimed invention.

What is claimed is:

1. In a client-server system in which Web content is delivered from multiple servers to a client, a client-based system implemented at the client comprising:

a scheduling subsystem to schedule times to obtain the Web content from the servers without the servers having prearranged knowledge of the times, whereupon

reaching a scheduled time, the scheduling subsystem generates an event notification containing information regarding how to retrieve the Web content from a corresponding server;

a delivery subsystem, responsive to the event notification, to retrieve the Web content and an index of the Web content from the corresponding server;

a cache to store the index and the Web content obtained by the delivery system;

an indexing subsystem to retrieve the index from the cache and present the index to a user, the indexing subsystem including a user interface which enables the user to select from the index portions of the Web content stored in the cache; and

a filter to condense the index according to preferences of the user.

2. A client-based system as recited in claim **1**, wherein the scheduling subsystem comprises:

a scheduler to schedule the time to obtain the Web content; and

a notification generator to generate the event notification at the scheduled time.

3. A client-based system as recited in claim **1**, wherein:

the delivery subsystem comprises multiple delivery modules that utilize different distribution systems to retrieve the index and the Web content; and

the event notification contains instructions dictating which of the delivery modules is to be used.

4. A client-based system as recited in claim **1**, wherein the Web content is multicast to a network address, the delivery subsystem comprising a listener program to listen to the network address at the scheduled time to retrieve the Web content.

5. A client-based system as recited in claim **1**, wherein the delivery subsystem comprises a fetching program to access the server and retrieve the Web content from the server.

6. A client-based system as recited in claim **1**, wherein the Web content is transmitted as a broadcast data stream over a broadcast medium from the server, the delivery subsystem being coupled to receive the broadcast data stream from a broadcast receiver and to reconstruct the Web content from the data stream.

7. A client-based system as recited in claim **1**, further comprising a content user interface to present the Web content to the user.

8. A Web browser application, embodied on a computer-readable medium, comprising:

computer-executable instructions to schedule a time to obtain Web content from a server without the server having prearranged knowledge of the scheduled time;

computer-executable instructions to generate an event notification upon occurrence of a scheduled time, the event notification containing information regarding how to retrieve the Web content;

computer-executable instructions to retrieve the Web content and an index of the Web content;

computer-executable instructions to present the index to a user and to enable the user to select certain Web content identified in the index; and

computer-executable instructions to filter the index according to user preferences.

9. A Web browser application as recited in claim **8**, further comprising computer-executable instructions to listen to a multicast address to retrieve at least one of the index and the Web content.

10. A Web browser application as recited in claim **8**, further comprising computer-executable instructions to access a remote server and retrieve at least one of the index and the Web content.

11. A system for delivering Web content over a medium, comprising:

a gathering subsystem located at a webcast center to gather Web content from sites on the Internet and to store the Web content;

a scheduling subsystem implemented at a client remote from the webcast center to schedule a time for the client to retrieve the Web content from the webcast server;

a delivery subsystem implemented at the client and responsive to the scheduling subsystem to obtain the Web content from the webcast center at the time set by the scheduling subsystem;

a program implemented at the client to cache a user's preferences regarding types of the Web content;

an indexing subsystem at the client to obtain an index of the Web content and present the index to a user, the indexing subsystem including a user interface which enables the user to select certain Web content identified in the index; and

a filter to filter the index according to the user's preferences.

12. A system as recited in claim **11**, further comprising:

a multicast transmitter at the webcast center to multicast the Web content to a multicast address; and

the delivery subsystem comprising a listener program to listen to the multicast address to retrieve the Web content.

13. A system as recited in claim **11**, wherein the delivery subsystem comprises means for accessing the server and retrieving the Web content from the server.

14. A system as recited in claim **11**, further comprising:

a broadcast transmitter to broadcast the Web content from the webcast center as a broadcast data stream over a broadcast medium; and

the delivery subsystem being coupled to receive the broadcast data stream from a broadcast receiver and to reconstruct the Web content from the data stream.

15. A system as recited in claim **11**, wherein the filter is implemented at the webcast center.

16. A system as recited in claim **11**, wherein the filter is implemented at the server.

17. A system as recited in claim **11**, wherein the webcast center maintains the index of the Web content and wherein the indexing subsystem obtains the index from the webcast center.

18. A system as recited in claim **11**, further comprising:

a cache implemented at the client;

a data submission subsystem implemented at the client to accumulate data, which is destined for the server for server-side processing, within the cache while the client is offline from the webcast center; and

the data submission subsystem submitting the data accumulated in the cache to the webcast center during an online session between the client and the server.

19. In a client-server system in which Web content is delivered from a server to a client, a computer-implemented method implemented at the client comprising the following steps:

scheduling a time to obtain the Web content from the server without the server having prearranged knowledge of the scheduled time;

listening to a multicast address to retrieve the Web content from the server at the scheduled time;

locally caching the Web content obtained from the server;

obtaining an index of the Web content from the server; and

filtering the index according to user preferences.

20. A computer-implemented method as recited in claim **11**, further comprising the step of caching the Web content according to a unique identification.

21. A computer-implemented method as recited in claim **11**, further comprising the step of caching the Web content according to a corresponding universal resource locator.

22. A computer-implemented method as recited in claim **19**, further comprising the following steps:

presenting the index to a user; and

enabling a user to select items in the index as an indication of preferred Web content.

23. A computer-implemented method as recited in claim **11**, further comprising the following steps:

caching data to be submitted to the server when the client is offline from the server; and

submitting the cached data to the server during an online session in which the client is actively connected to the server.

24. A computer-implemented method as recited in claim **11**, further comprising the following steps:

enabling a user to select Web content obtained from different channels; and

aggregating the Web content into a single channel for presentation to the user.

25. A computer-readable medium having computer-executable instructions for performing the steps as recited in claim **11**.

26. A browser application, embodied on a computer-readable medium, having computer-executable instructions for performing the steps as recited in claim **11**.

* * * * *

US006473855B1

(12) **United States Patent**
Welder

(10) **Patent No.: US 6,473,855 B1**
(45) **Date of Patent: Oct. 29, 2002**

(54) **METHOD AND APPARATUS FOR PROVIDING CONTENT ON A COMPUTER SYSTEM BASED ON USAGE PROFILE**

(75) Inventor: **W. Dean Welder**, Boulder Creek, CA (US)

(73) Assignee: **Phoenix Technologies Ltd.**, San Jose, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/336,111**

(22) Filed: **Jun. 18, 1999**

(51) **Int. Cl.**[7] **G06F 13/14**; G06F 13/20; G06F 1/04

(52) **U.S. Cl.** **713/2**; 710/29; 710/33; 710/34; 713/502

(58) **Field of Search** 710/33, 29, 34; 713/2, 502

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,121,345	A		6/1992	Lentz
5,128,995	A		7/1992	Arnold et al.
5,131,089	A		7/1992	Cole
5,142,680	A		8/1992	Ottman et al.
5,146,568	A		9/1992	Flaherty et al.
5,212,798	A	*	5/1993	Kanda 395/775
5,214,695	A		5/1993	Arnold et al.
5,274,816	A		12/1993	Oka
5,280,627	A		1/1994	Flaherty et al.
5,307,497	A		4/1994	Feigenbaum et al.
5,325,532	A		6/1994	Crosswy et al.
5,379,431	A		1/1995	Lemon et al.
5,381,549	A		1/1995	Tamura
5,418,918	A		5/1995	Vander Kamp et al.
5,444,850	A		8/1995	Chang
5,448,741	A		9/1995	Oka
5,452,454	A		9/1995	Basu
5,463,766	A		10/1995	Schieve et al.

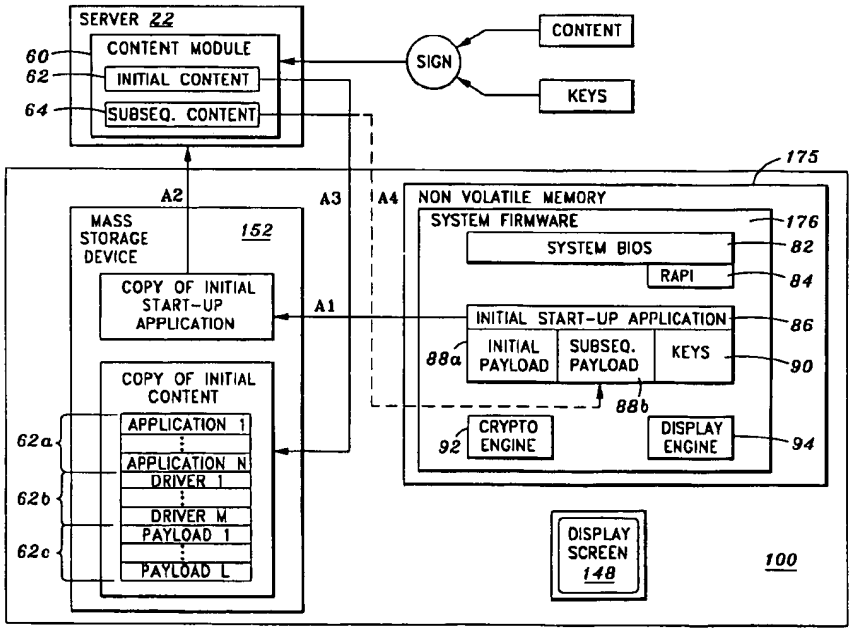
(List continued on next page.)

Primary Examiner—Jeffrey Gaffin
Assistant Examiner—Rehana Penveen

(57) **ABSTRACT**

A method and apparatus for determining a computer system usage profile, and transmitting the computer system usage profile to a server which targets content to the computer system in response to the usage profile is described. A basic input output system (BIOS) module and/or an operating system module obtain computer system usage profile information by tracking events such as the frequency of re-boots, the time required to boot-up and shut-down the operating system on the computer system, the amount of time the computer system is "used", and the frequency and amount of time the computer system is connected to the Internet. This data is collected and communicated to a profile server. The profile server targets content such as messages with graphics or informational material, etc. to the computer system based upon the computer system usage profile. In one embodiment, the content is displayed during boot-up and shut-down of the operating system.

17 Claims, 10 Drawing Sheets

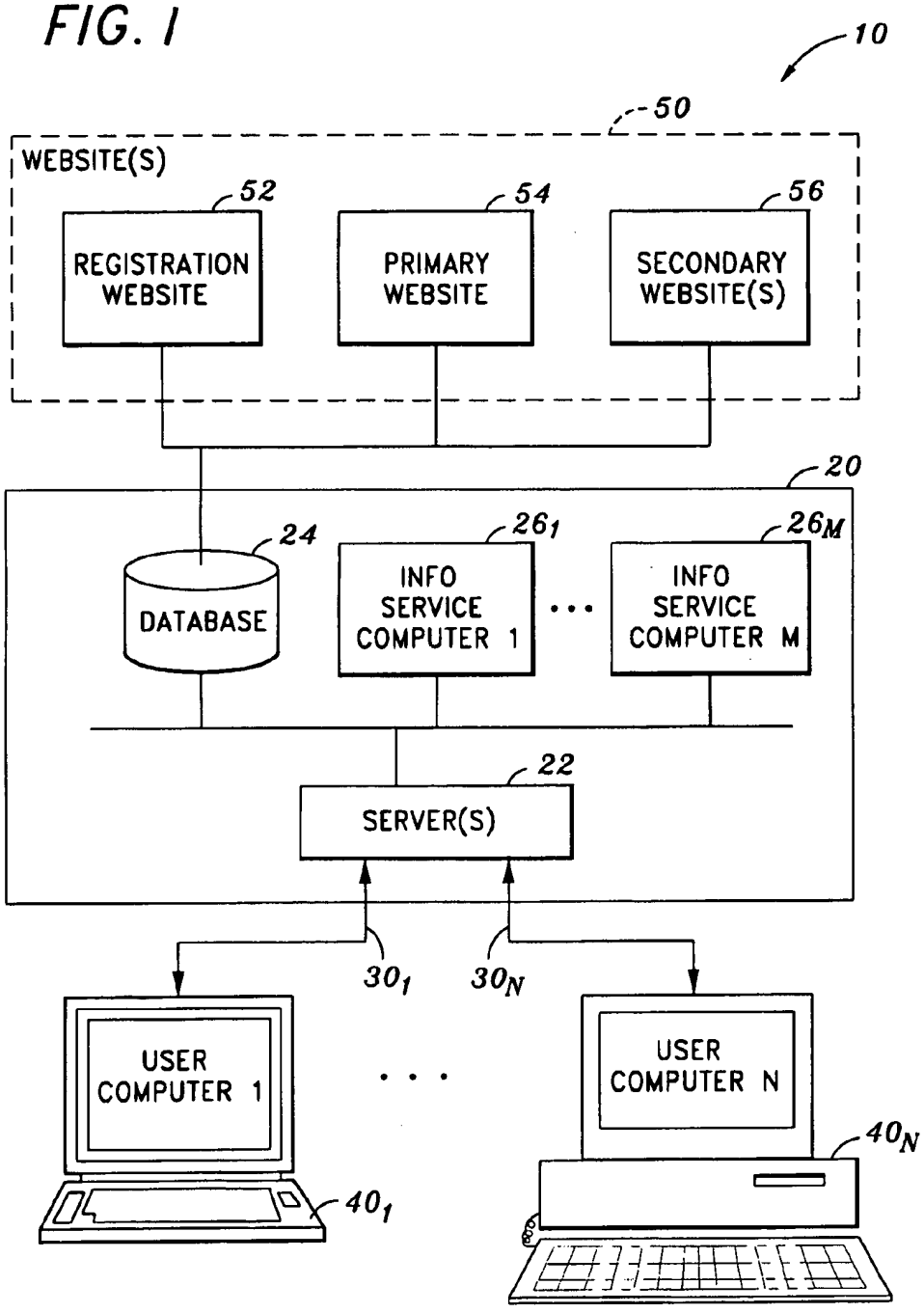


U.S. PATENT DOCUMENTS

5,469,573	A		11/1995	McGill, III et al.
5,504,905	A		4/1996	Cleary et al.
5,522,076	A		5/1996	Dewa et al.
5,526,523	A		6/1996	Straub et al.
5,542,082	A		7/1996	Solhjell
5,581,740	A		12/1996	Jones
5,586,327	A		12/1996	Bealkowski et al.
5,594,903	A		1/1997	Bunnell et al.
5,604,890	A		2/1997	Miller
5,652,868	A		7/1997	Williams
5,652,886	A		7/1997	Tulpule et al.
5,664,194	A		9/1997	Paulsen
5,680,547	A		10/1997	Chang
5,692,190	A		11/1997	Williams
5,694,583	A		12/1997	Williams et al.
5,694,600	A		12/1997	Khenson et al.
5,701,477	A		12/1997	Chejlava, Jr.
5,715,456	A		2/1998	Bennett et al.
5,717,930	A		2/1998	Imai et al.
5,727,213	A		3/1998	Vander Kamp et al.
5,732,268	A		3/1998	Bizzarri
5,748,957	A		5/1998	Klein
5,754,853	A		5/1998	Pearce
5,764,593	A		6/1998	Turpin et al.
5,768,627	A	*	6/1998	Jones et al. 395/880
5,781,758	A		7/1998	Morley
5,790,849	A		8/1998	Crocker et al.
5,796,984	A		8/1998	Pearce et al.
5,802,363	A		9/1998	Williams et al.
5,805,880	A		9/1998	Pearce et al.
5,805,882	A		9/1998	Cooper et al.
5,815,706	A		9/1998	Stewart et al.
5,819,063	A		10/1998	Dahl et al.
5,828,888	A		10/1998	Kozaki et al.
5,832,251	A		11/1998	Takahashi
5,842,011	A		11/1998	Basu
5,854,905	A		12/1998	Garney
5,864,698	A		1/1999	Krau et al.
5,887,164	A		3/1999	Gupta
5,901,310	A		5/1999	Rahman et al.
5,907,679	A		5/1999	Hoang et al.
5,920,896	A	*	7/1999	Grimsrud et al. 711/165
6,138,234	A	*	10/2000	Lee et al. 713/2
6,202,190	B1	*	3/2001	Rogier 714/815

* cited by examiner

FIG. 1


10
50
WEBSITE(S)
52
REGISTRATION WEBSITE
54
PRIMARY WEBSITE
56
SECONDARY WEBSITE(S)
20
24
DATABASE
26₁
INFO SERVICE COMPUTER 1
26M
INFO SERVICE COMPUTER M
22
SERVER(S)
30₁
30N
USER COMPUTER 1
USER COMPUTER N
40₁
40N

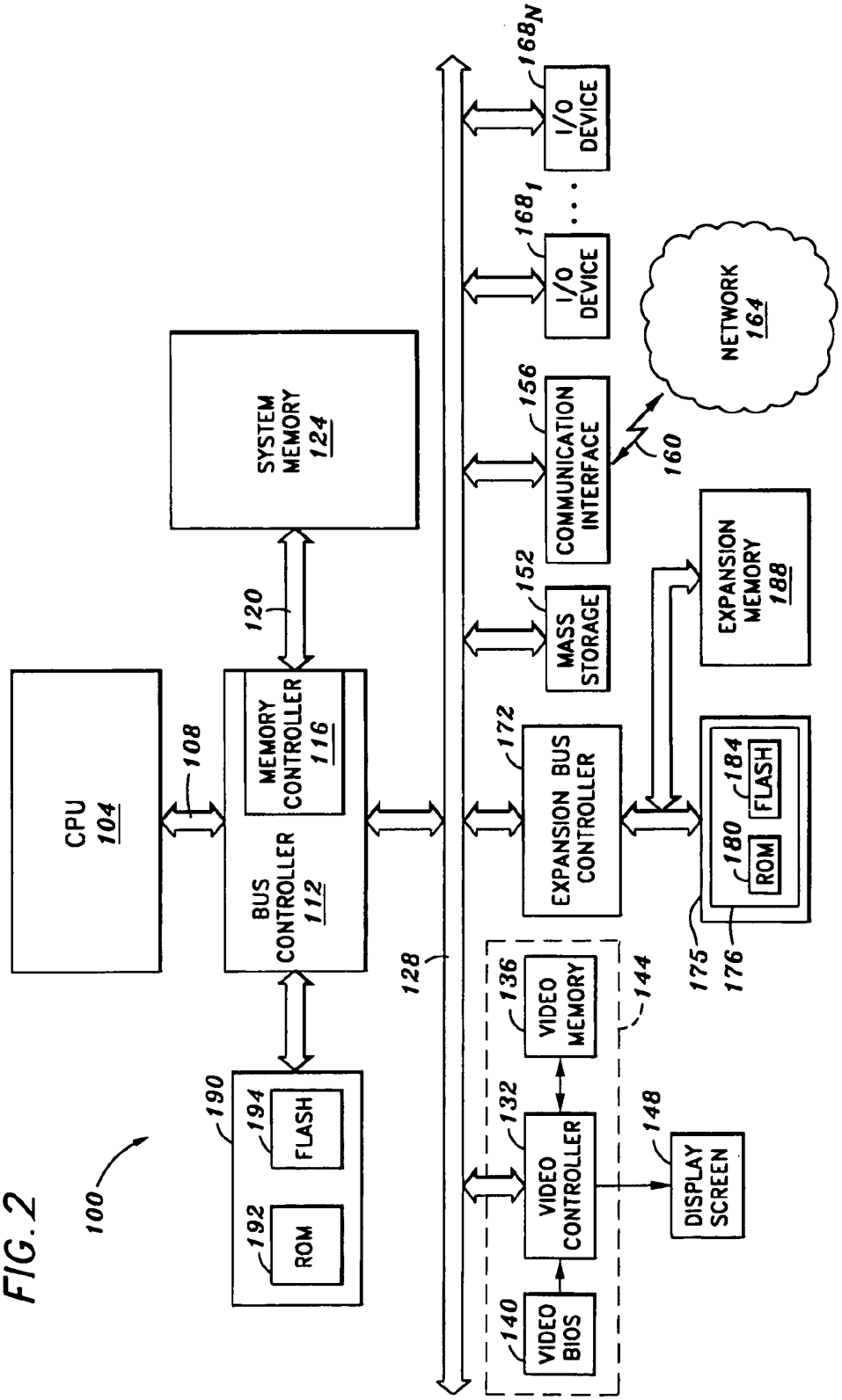
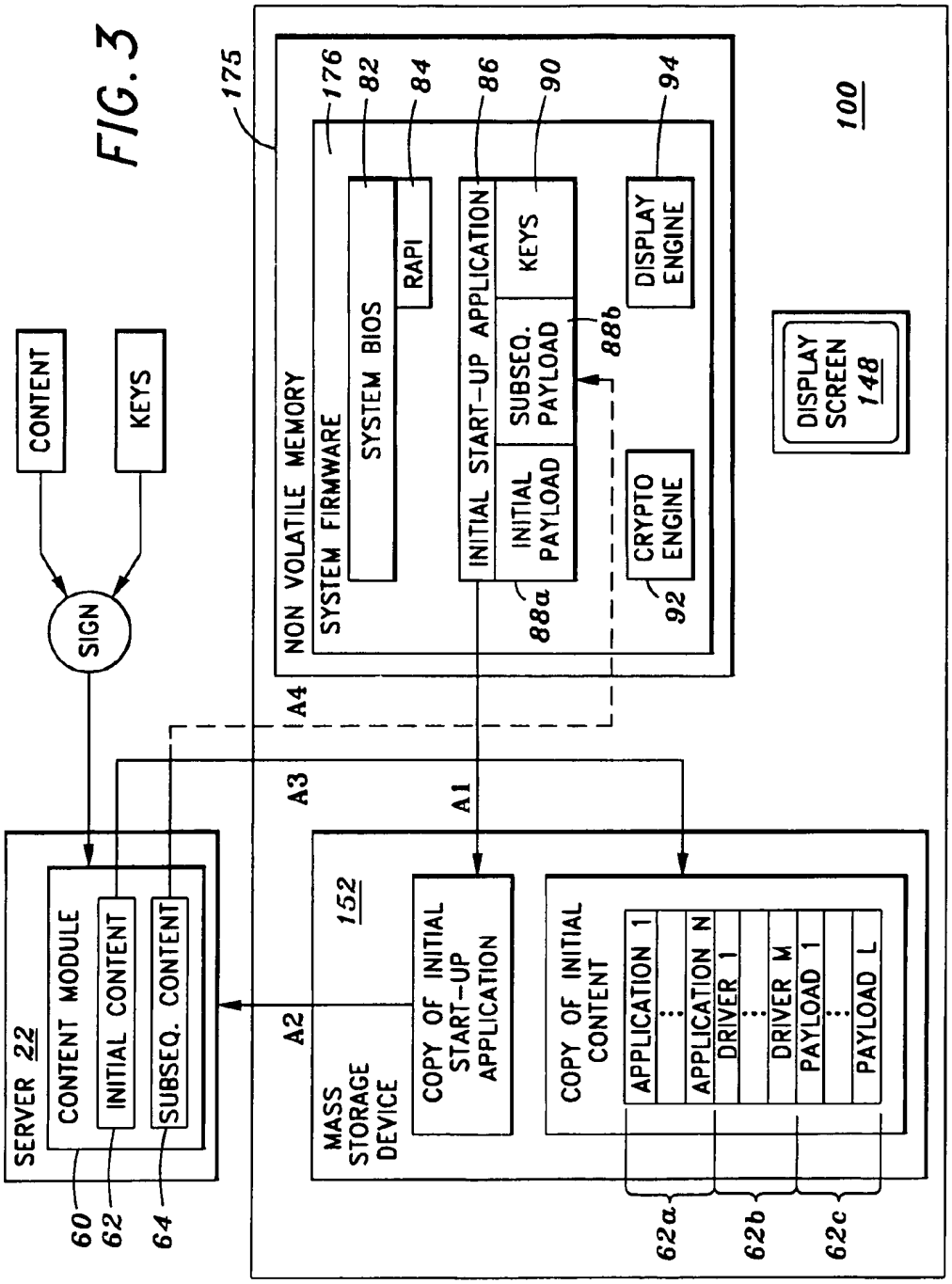

100
CPU 104
108
BUS CONTROLLER 112
MEMORY CONTROLLER 116
120
SYSTEM MEMORY 124
128
190
194
FLASH
192
ROM
136
VIDEO MEMORY
144
132
VIDEO CONTROLLER
148
DISPLAY SCREEN
140
VIDEO BIOS
172
EXPANSION BUS CONTROLLER
152
MASS STORAGE
156
COMMUNICATION INTERFACE
160
NETWORK 164
168N
I/O DEVICE
1681
I/O DEVICE
EXPANSION MEMORY 188
180
ROM
184
FLASH
175
176

FIG. 2

FIG. 3


100
175
NON VOLATILE MEMORY
176
SYSTEM FIRMWARE
82
SYSTEM BIOS
84
RAPI
86
INITIAL START-UP APPLICATION
88a
INITIAL PAYLOAD
88b
SUBSEQ. PAYLOAD
90
KEYS
92
CRYPTO ENGINE
94
DISPLAY ENGINE
DISPLAY SCREEN 148
CONTENT
KEYS
SIGN
A1
A2
A3
A4
SERVER 22
CONTENT MODULE
INITIAL CONTENT
SUBSEQ. CONTENT
60
62
64
152
MASS STORAGE DEVICE
COPY OF INITIAL START-UP APPLICATION
COPY OF INITIAL CONTENT
APPLICATION 1
APPLICATION N
DRIVER 1
DRIVER M
PAYLOAD 1
PAYLOAD L
62a
62b
62c

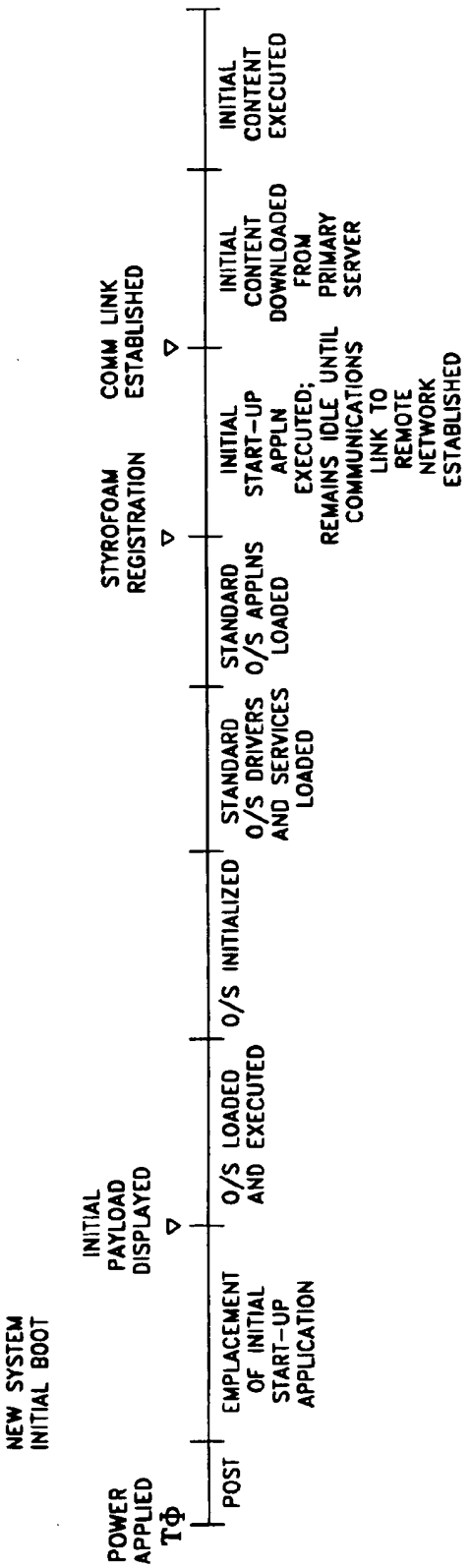

NEW SYSTEM INITIAL BOOT
POWER APPLIED
TΦ
POST
EMPLACEMENT OF INITIAL START-UP APPLICATION
INITIAL PAYLOAD DISPLAYED
O/S LOADED AND EXECUTED
O/S INITIALIZED
STANDARD O/S DRIVERS AND SERVICES LOADED
STANDARD O/S APPLNS LOADED
STYROFOAM REGISTRATION
INITIAL START-UP APPLN EXECUTED; REMAINS IDLE UNTIL COMMUNICATIONS LINK TO REMOTE NETWORK ESTABLISHED
COMM LINK ESTABLISHED
INITIAL CONTENT DOWNLOADED FROM PRIMARY SERVER
INITIAL CONTENT EXECUTED
INFOMEDIARY DRIVERS INSTALLED
INFOMEDIARY APPLNS INSTALLED
APPLNS AND DRIVERS EXECUTED; PROFILE MGR SENDS USER AND HARDWARE PROFILE TO SERVER CONTINUOUSLY UNTIL POWER DOWN
PAYLOAD 1-N DOWNLOADED
MEMORY UPDATED
END

FIG. 4A

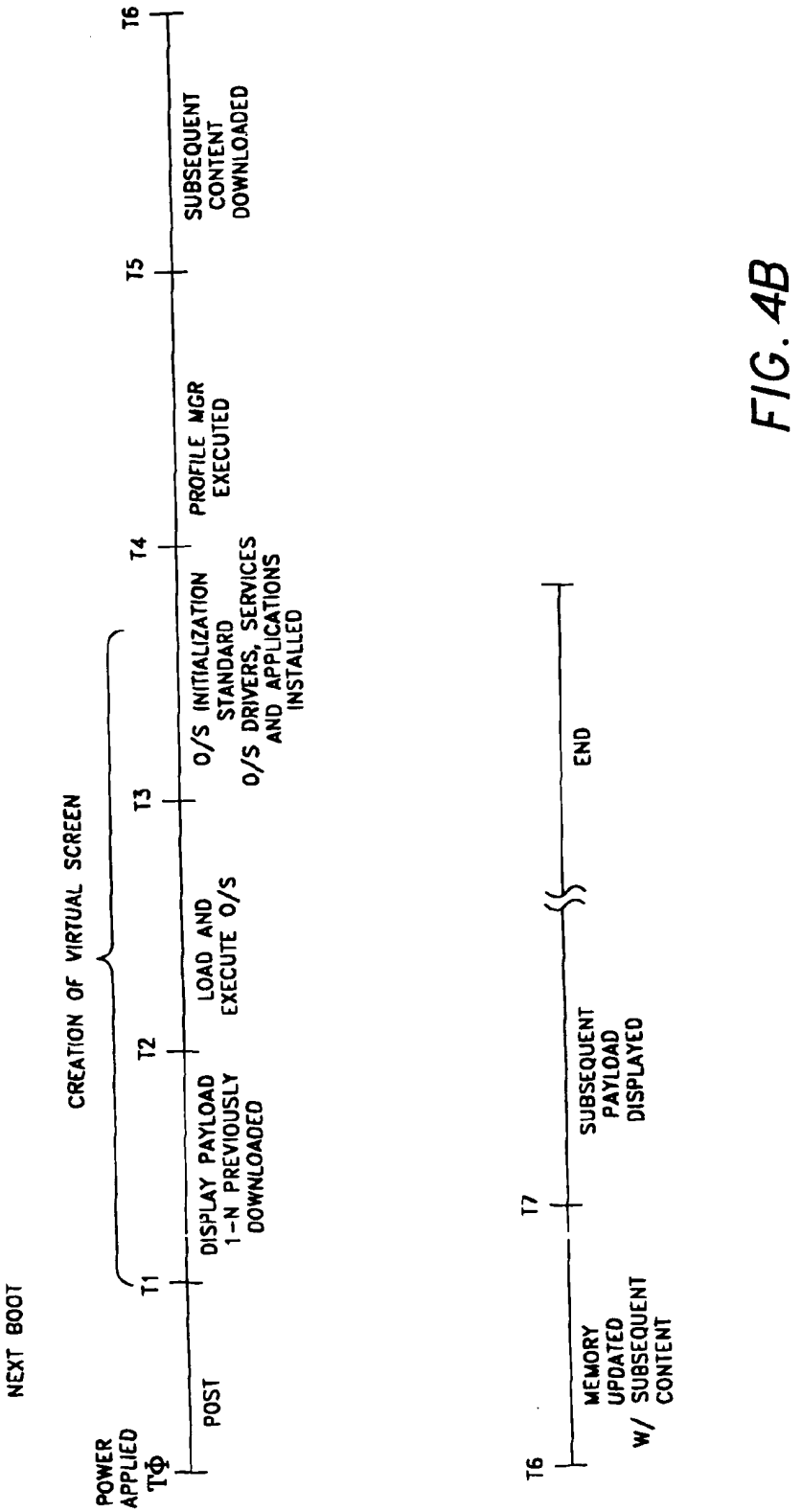

POWER APPLIED
TΦ
POST
T1
DISPLAY PAYLOAD 1-N PREVIOUSLY DOWNLOADED
T2
LOAD AND EXECUTE O/S
T3
O/S INITIALIZATION STANDARD O/S DRIVERS, SERVICES AND APPLICATIONS INSTALLED
T4
PROFILE MGR EXECUTED
T5
SUBSEQUENT CONTENT DOWNLOADED
T6
CREATION OF VIRTUAL SCREEN
NEXT BOOT
T6
MEMORY UPDATED W/ SUBSEQUENT CONTENT
T7
SUBSEQUENT PAYLOAD DISPLAYED
END

FIG. 4B

FIG. 5

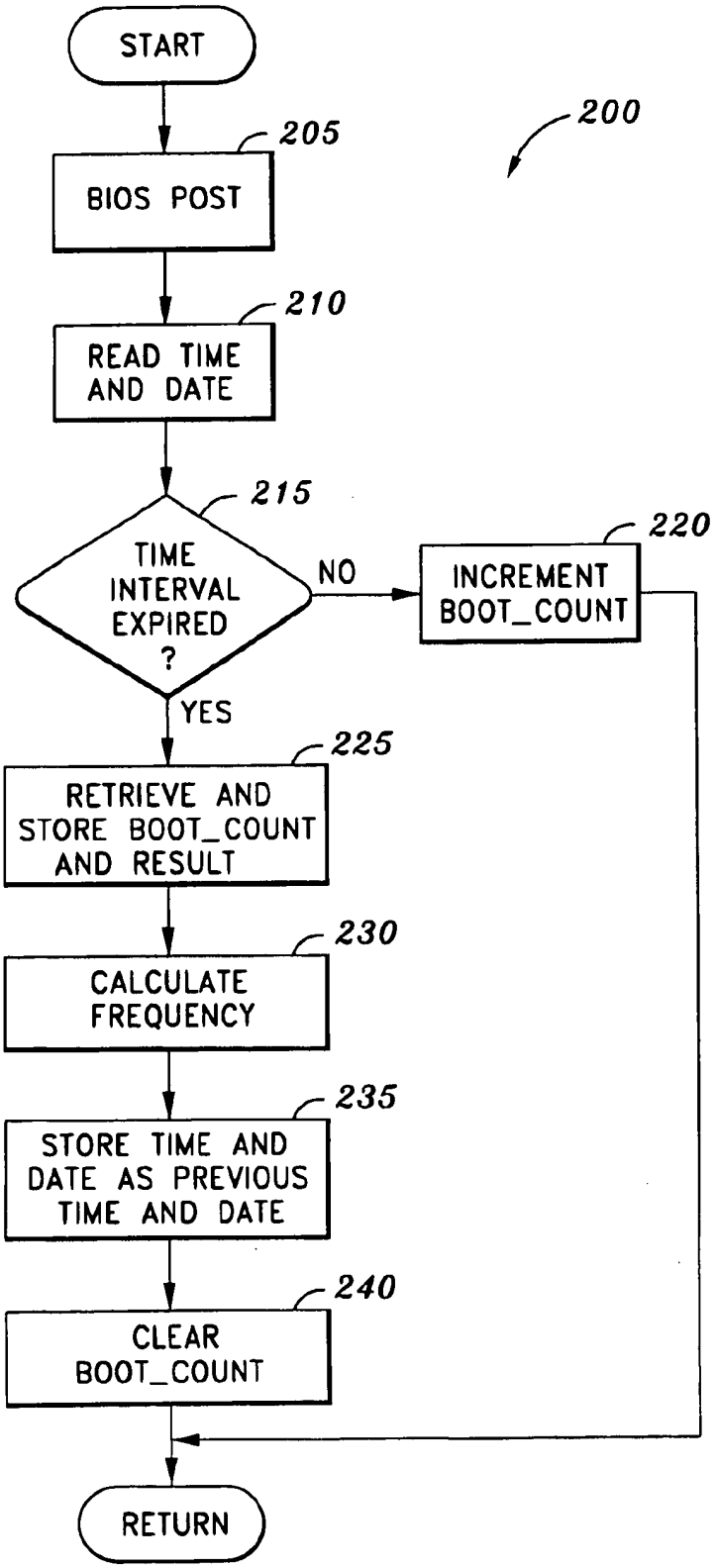

START
200
205
BIOS POST
210
READ TIME AND DATE
215
TIME INTERVAL EXPIRED ?
NO
220
INCREMENT BOOT_COUNT
YES
225
RETRIEVE AND STORE BOOT_COUNT AND RESULT
230
CALCULATE FREQUENCY
235
STORE TIME AND DATE AS PREVIOUS TIME AND DATE
240
CLEAR BOOT_COUNT
RETURN

FIG. 6

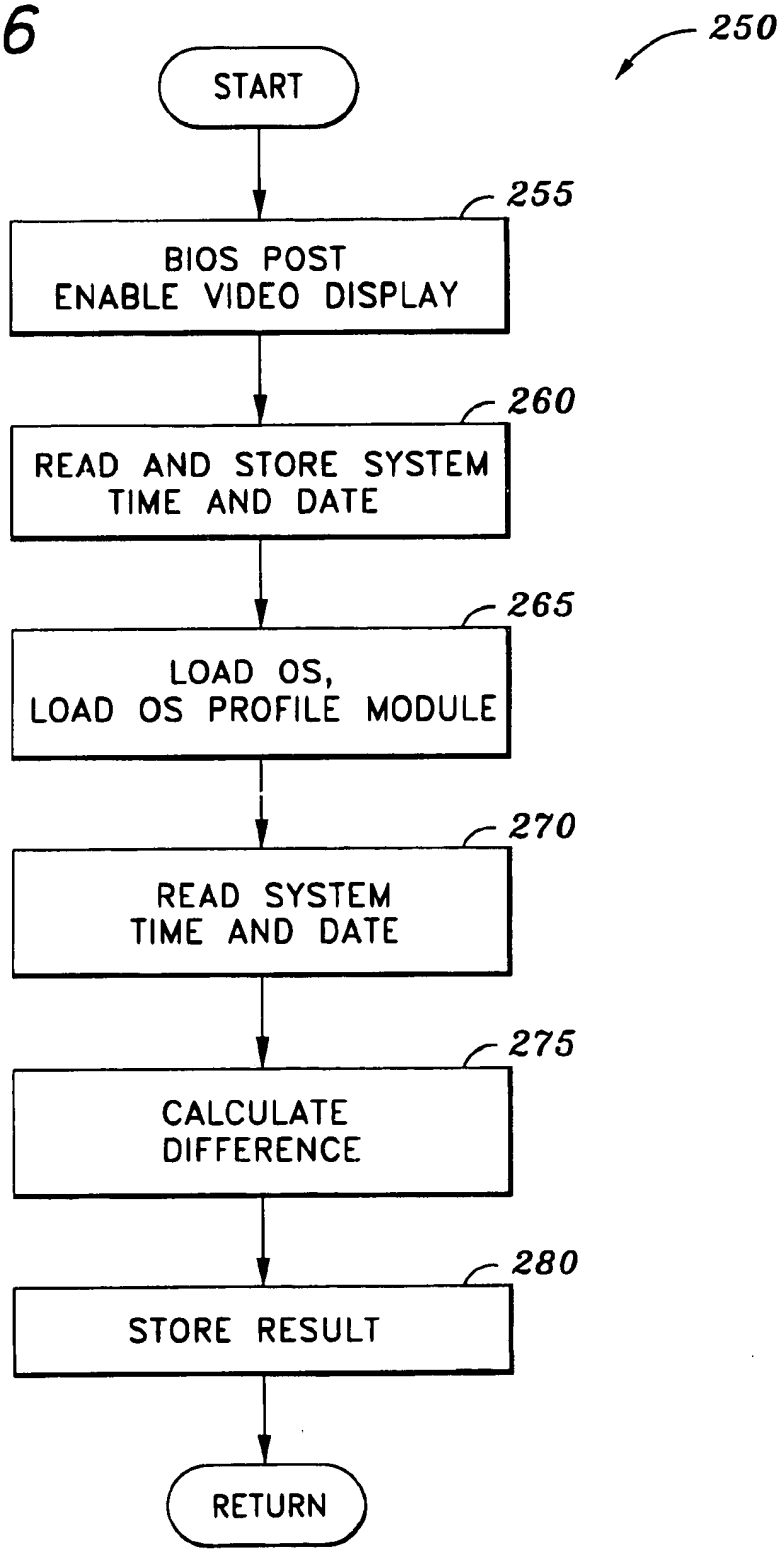

250
START
255
BIOS POST
ENABLE VIDEO DISPLAY
260
READ AND STORE SYSTEM
TIME AND DATE
265
LOAD OS,
LOAD OS PROFILE MODULE
270
READ SYSTEM
TIME AND DATE
275
CALCULATE
DIFFERENCE
280
STORE RESULT
RETURN

FIG. 7

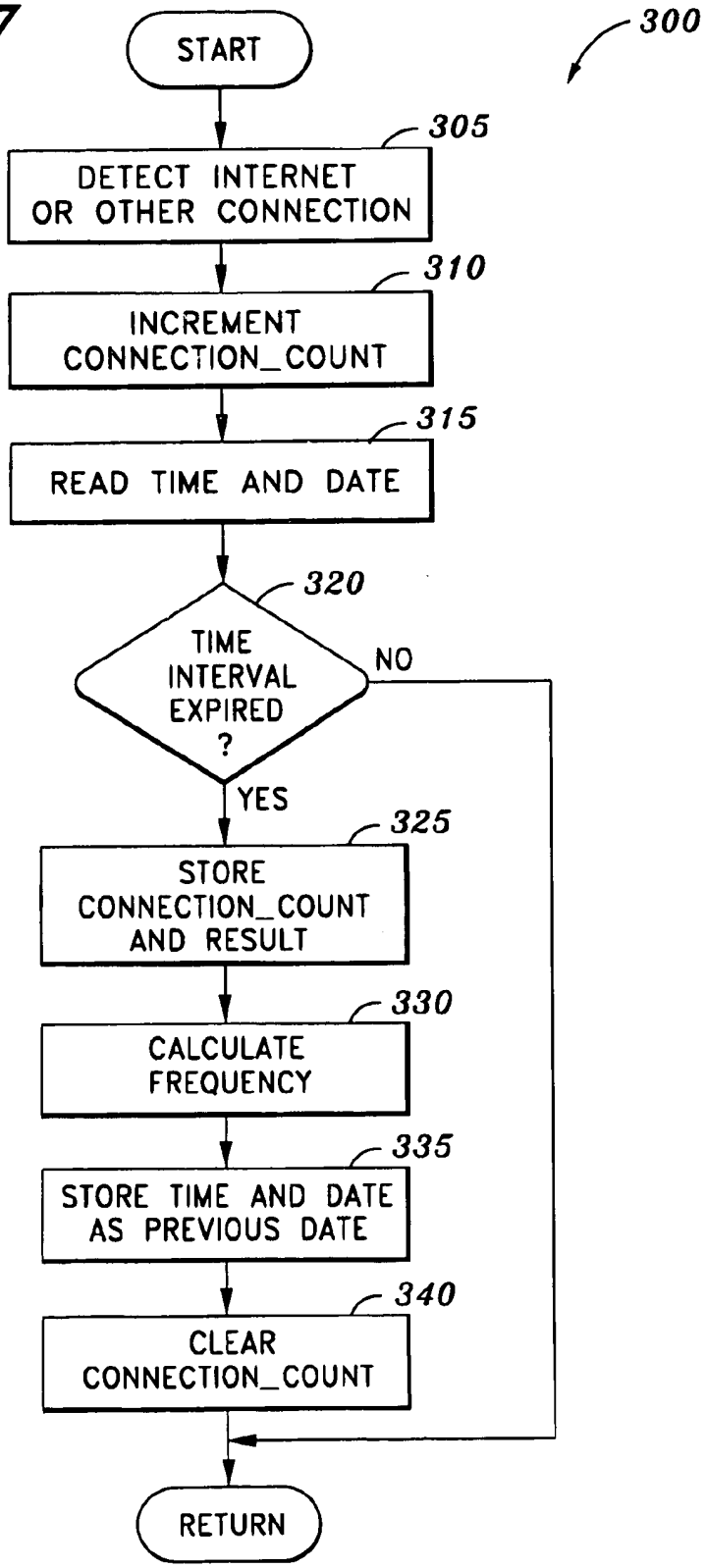

300
START
305
DETECT INTERNET OR OTHER CONNECTION
310
INCREMENT CONNECTION_COUNT
315
READ TIME AND DATE
320
TIME INTERVAL EXPIRED ?
NO
YES
325
STORE CONNECTION_COUNT AND RESULT
330
CALCULATE FREQUENCY
335
STORE TIME AND DATE AS PREVIOUS DATE
340
CLEAR CONNECTION_COUNT
RETURN

FIG. 8

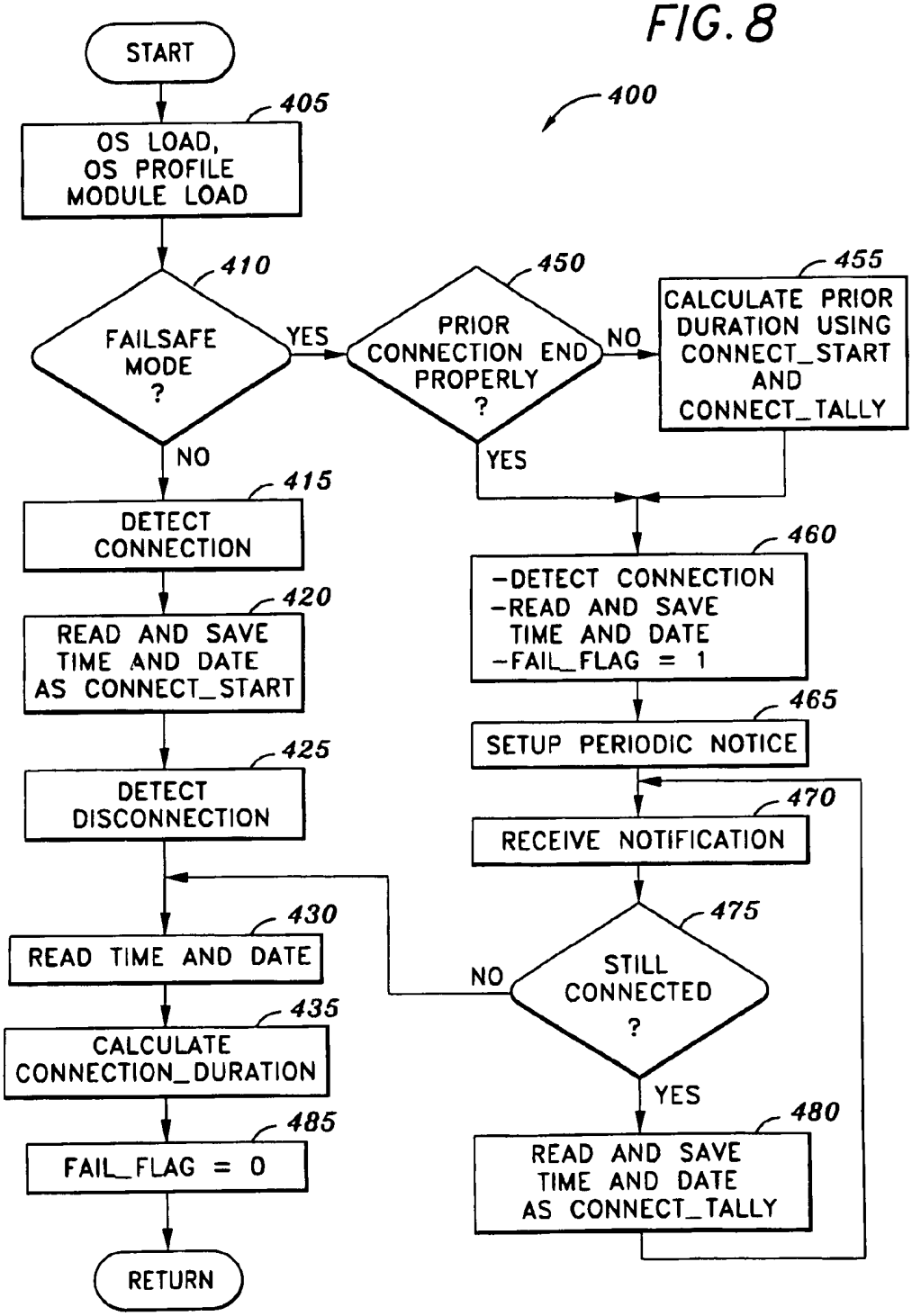
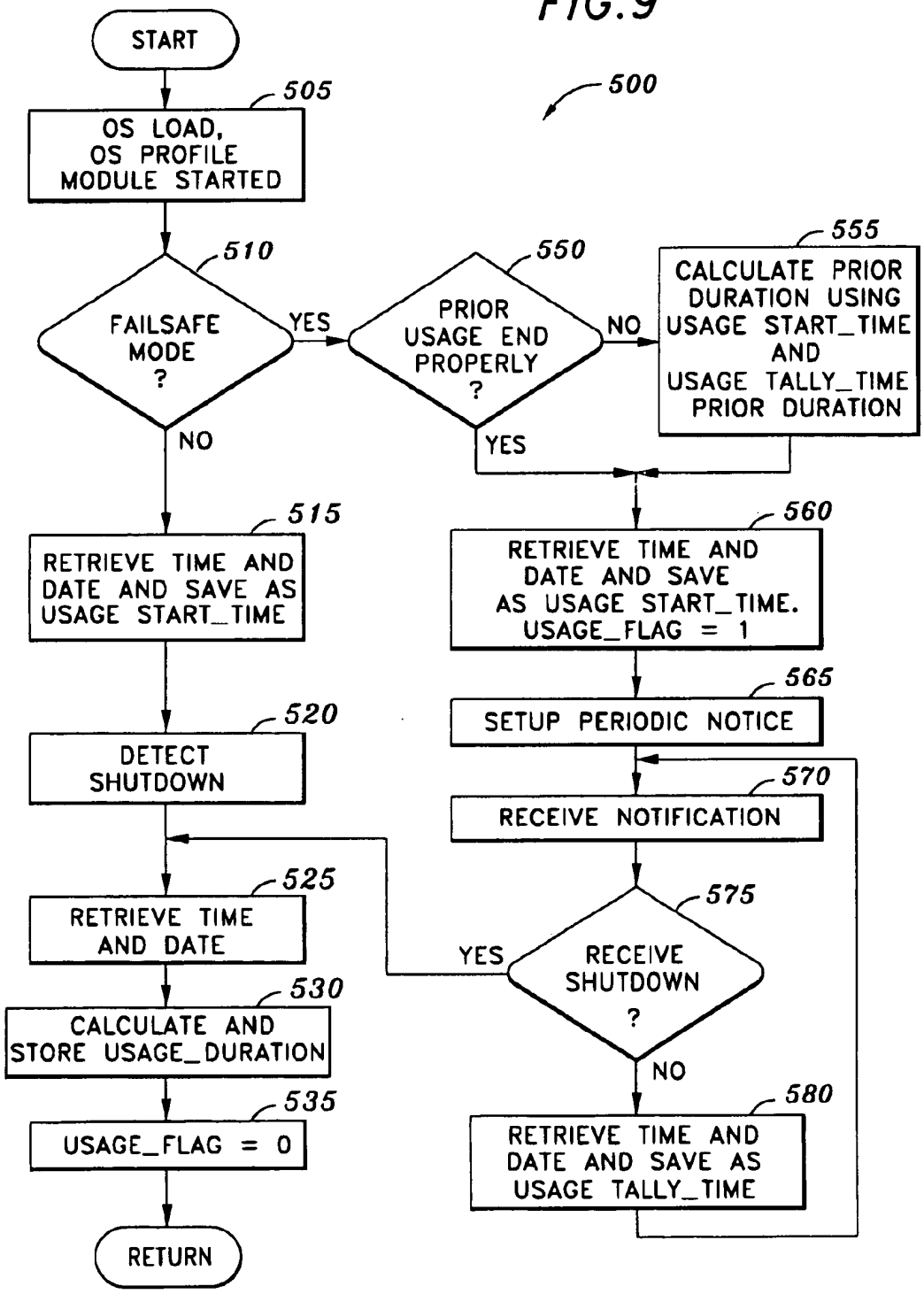

START
400
405
OS LOAD, OS PROFILE MODULE LOAD
410
FAILSAFE MODE ?
YES
450
PRIOR CONNECTION END PROPERLY ?
NO
455
CALCULATE PRIOR DURATION USING CONNECT_START AND CONNECT_TALLY
NO
YES
415
DETECT CONNECTION
420
READ AND SAVE TIME AND DATE AS CONNECT_START
425
DETECT DISCONNECTION
460
-DETECT CONNECTION
-READ AND SAVE TIME AND DATE
-FAIL_FLAG = 1
465
SETUP PERIODIC NOTICE
470
RECEIVE NOTIFICATION
475
STILL CONNECTED ?
NO
YES
430
READ TIME AND DATE
435
CALCULATE CONNECTION_DURATION
480
READ AND SAVE TIME AND DATE AS CONNECT_TALLY
485
FAIL_FLAG = 0
RETURN

FIG. 9


500
START
505
OS LOAD, OS PROFILE MODULE STARTED
510
FAILSAFE MODE ?
YES
NO
550
PRIOR USAGE END PROPERLY ?
NO
YES
555
CALCULATE PRIOR DURATION USING USAGE START_TIME AND USAGE TALLY_TIME PRIOR DURATION
515
RETRIEVE TIME AND DATE AND SAVE AS USAGE START_TIME
560
RETRIEVE TIME AND DATE AND SAVE AS USAGE START_TIME. USAGE_FLAG = 1
565
SETUP PERIODIC NOTICE
520
DETECT SHUTDOWN
570
RECEIVE NOTIFICATION
525
RETRIEVE TIME AND DATE
575
RECEIVE SHUTDOWN ?
YES
NO
530
CALCULATE AND STORE USAGE_DURATION
535
USAGE_FLAG = 0
580
RETRIEVE TIME AND DATE AND SAVE AS USAGE TALLY_TIME
RETURN

METHOD AND APPARATUS FOR PROVIDING CONTENT ON A COMPUTER SYSTEM BASED ON USAGE PROFILE

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates generally to the field of computer systems, and specifically, to a method and apparatus for determining computer system usage profile information.

2. Background Information

In a typical computer, such as a PC-compatible computer, the boot process involves performing various system tests and displaying a basic input-output system (BIOS) information screen. This time period is referred to as power on self test (POST). Once the BIOS completes the POST, it loads a native operating system. The operating system then replaces the BIOS information screen with the operating system's own display screen. The operating system, such as for example, the Windows™ operating system, displays its own proprietary splash or "cloud screen" while the operating system is loading. The display screen shows a static, graphical company logo and product image with an activity indicator. The activity indicator generally shows activity near the bottom of the screen either using color manipulation or presenting a progress bar with color filling to indicate the current progress of the operating system loading.

During the time the operating system is loading, there is no informational content displayed on the screen for the user. Moreover, the time spent loading the operating system is significantly longer then the POST of the BIOS. Consequently, a much shorter duration of time is given to the user to view and read the contents of the BIOS information screen display or other content for viewing by users.

During the time the operating system is loading and/or shutting down, content such as messages with graphics or informational material can be displayed on the display screen for the user. The longer the operating system takes to load and/or shut down, the more content can be displayed on the display screen.

SUMMARY OF THE INVENTION

The present invention is a method of determining boot-up time of a computer system. In one embodiment, the method includes retrieving a first time on the computer system, loading the operating system, retrieving a second time on the computer system when the operating system has loaded, and determining a boot time in response to the first and second times.

Other embodiments are described and claimed herein.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a system block diagram of one embodiment of an information distribution system in which the apparatus and method of the invention is used.

FIG. 2 illustrates an exemplary processor system or user computer system which implements embodiments of the present invention.

FIG. 3 illustrates a logical diagram of one embodiment of the invention.

FIGS. 4A and 4B illustrate one embodiment of a process flow chart provided in accordance with the principles of the invention.

FIG. 5 shows an exemplary flow diagram of a portion of a BIOS profile module, according to one embodiment of the present invention.

FIG. 6 shows a flow diagram showing a process for determining boot duration, according to one embodiment of the present invention.

FIG. 7 shows an exemplary flow diagram of a portion of the OS profile module, according to one embodiment of the present invention.

FIG. 8 shows an exemplary flow diagram of another portion of the OS profile module, according to one embodiment of the present invention

FIG. 9 shows an exemplary flow diagram of yet another portion of the OS profile module, according to one embodiment of the present invention

DETAILED DESCRIPTION

The present invention comprises a method and apparatus for determining computer system usage profile on a computer system, and transmitting the computer system usage profile to a server which targets content to the computer system in response to the usage profile. The present invention includes a basic input output system (BIOS) module and/or an operating system module which obtain computer system usage profile information. The modules track items such as the frequency of re-boots, the time required to boot and shut-down the operating system on the computer system, the amount of time the computer system is "used", and the frequency and amount of time the computer system is connected to a network (e.g., the Internet). This data is collected and communicated to a profile server. The profile server targets content such as messages with graphics or informational material, etc. to the computer system based upon the usage profile. In one embodiment, the content is displayed during the time the operating system is loading and/or shutting down.

Definitions

As discussed herein, a "computer system" is a product including circuitry capable of processing data. The computer system may include, but is not limited to, general purpose computer systems (e.g., server, laptop, desktop, palmtop, personal electronic devices, etc.), personal computers (PCs), hard copy equipment (e.g., printer, plotter, fax machine, etc.), banking. equipment (e.g., an automated teller machine), and the like. An infomediary is a web site that provides information on behalf of producers of goods and services, supplying relevant information to businesses about products and/or services offered by suppliers and other businesses. Content refers to application programs, driver programs, utility programs, the payload, etc., and combinations thereof, as well as graphics, informational material (articles, stock quotes, etc.) and the like, either singly or in any combination. "Payload" refers to messages with graphics or informational material (such as articles, stock quotes, etc.) and may include files or applications. In one embodiment, it is transferred at a predetermined time to the system's mass storage media. In addition, a "communication link" refers to the medium or channel of communication. The communication link may include, but is not limited to, a telephone line, a modem connection, an Internet connection, an Integrated Services Digital Network ("ISDN") connection, an Asynchronous Transfer Mode (ATM) connection, a frame relay connection, an Ethernet connection, a coaxial connection, a fiber optic connection, satellite connections (e.g. Digital Satellite Services, etc.),

wireless connections, radio frequency (RF) links, electromagnetic links, two way paging connections, etc., and combinations thereof.

In addition, the loading of an operating system ("OS") refers to the initial placement of the operating system bootstrap loader. In one embodiment, during the OS load, a sector of information is typically loaded from a hard disk into the system memory. Alternatively, the bootstrap loader is loaded from a network into system memory. An OS "boot" refers to the execution of the bootstrap loader. This places the OS in control of the system. Some of the actions performed during the OS boot include system configuration, device detection, loading of drivers and user logins. OS runtime refers to the completion of the boot phase and the beginning of the execution of applications by the OS. In one embodiment, during OS runtime, the OS interacts with the user to execute and/ or run applications. Power On Self Test (POST) refers to the instructions that are executed to configure and test the system hardware prior to loading an OS.

System Overview

A description of an exemplary system, which incorporates embodiments of the present invention, is hereinafter described.

FIG. **1** shows a system block diagram of one embodiment of an information distribution system **10** in which the apparatus and method of the invention is used. The system **10** relates to providing an infomediary. It involves the construction and maintenance of a secure and private repository of Internet user and system profiles, collected primarily from warranty service registrations, Internet service registrations, system profiles, and user preferences. Initially, this information is used to register the user with the manufacturers of purchased hardware and software products, and with the providers of on-line or other services. Over time, the user data is used to create a user profile and notify users of relevant software updates and upgrades, to encourage on-line purchases of related products, and to enable one-to-one customized marketing and other services.

In one embodiment, two software modules are used to implement various embodiments of the invention. One is resident on a user's system, and is used to access a predetermined web site. For example, in one embodiment, the operating system and Basic Input and Output System (BIOS) are pre-installed on a computer system, and when the computer system is subsequently first powered up, an application, referred to for discussion purposes as the first software module (in one embodiment, the first software module is the initial start-up application (ISUA), which will be described in the following sections), will allow the launching of one or more executable programs in the pre-boot environment. In one embodiment, the first software module facilitates the launching of one or more executable programs prior to the loading, booting, execution and/or running of the OS. In one embodiment, the user is encouraged to select the use of such a program (i.e., the use of the first software module), and in alternative embodiments, the program is automatically launched. The program(s) contained in the first software module enables tools and utilities to run at an appropriate time, and with proper user authorization, also allow the user to download a second software module that includes drivers, applications and additional payloads through the Internet connection on the PC. The programs may also provide for remote management of the system if the OS fails to launch successfully.

Once the second software module has been delivered, it may become memory resident, and may disable the transferred copy of the first software module. The original copy of the first software module still residing in the system's non-volatile memory remains idle until the-second software module fails to function, becomes corrupted or is deleted, upon which a copy of the original first software module is again transferred as described above. The second software module may include an application that connects the user to a specific server on the Internet and directs the user to a predetermined web site to seek authorization to down load further subscription material. The second software module may also include content that is the same or similar to the content of the first software module.

In one embodiment, the system may also include an initial payload that is stored in Read Only Memory BIOS (ROM BIOS). In one embodiment, the initial payload is part of the first software module (e.g., the ISUA). In an alternative embodiment, the initial payload is stored as a module in ROM BIOS, separate from the first software module. In one embodiment, the initial payload is launched from ROM BIOS and displayed on the screen after the Power On Self Test (POST) but prior to the booting, loading and/or execution of the OS. This may occur at a predetermined time, such as when the system is being manufactured, assembled and tested, or when the end user first activates the system. In an alternate embodiment, this initial payload is copied to a predetermined location (such as the system's hard disk) at a predetermined time, such as when the system is being manufactured, assembled and tested, or when the end user first activates the system. Once copied, the payload executes after POST but prior to operation of the OS, and may display graphics, messages with graphics or informational material, animation, Joint Photographic Experts Group (JPEG)/ Moving Picture Experts Group (MPEG) formatted material on the screen. When additional programs and/or payloads are delivered. (via the Internet or other outside connection), the display screen may be used to provide customized screens in the form of messages or graphics prior to and during booting of the OS. In addition, executable programs delivered in the first software module, as well as subsequent programs (such as the second software module) downloaded from the web site, may be used to survey the PC to determine various types of devices, drivers, and applications installed. In one embodiment, as described in co-pending U.S. patent application Ser. No. 09/336,289 entitled "Method and Apparatus for Automatically Installing And Configuring Software on a Computer" incorporated herein by reference, the first software. module is used to identify and to automatically create shortcuts and/or bookmarks for the user. The programs downloaded from the website may include software that collects and maintains a user profile based on the user's preferences. Such information may be provided to the infomediary, which subsequently forwards portions of the information and/or compiled data based on the information to suppliers and other businesses to obtain updates or revisions of information provided by the suppliers and other businesses.

Referring to FIG. **1**, the information distribution system **10** comprises a service center **20** that is connected over one or more communications links $\mathbf{30}_1$–$\mathbf{30}_N$ to one or more user computer systems $\mathbf{40}_1$–$\mathbf{40}_N$ ("40"). The service center **20** includes one or more servers **22**, one or more databases **24**, and one or more computers $\mathbf{26}_1\mathbf{26}_M$. The one or more computers $\mathbf{26}_1$–$\mathbf{26}_M$ are capable of simultaneous access by a plurality of the user computer systems $\mathbf{40}_1$–$\mathbf{4}_N$. If a plurality of computers are used, then the computers $\mathbf{26}_1$–$\mathbf{26}_M$ may be connected by a local area network (LAN) or any other similar connection technology. However, it is also possible

for the service center **20** to have other configurations. For example, a smaller number of larger computers (i.e. a few mainframe, mini, etc. computers) with a number of internal programs or processes running on the larger computers capable of establishing communications links to the user computers.

The service center **20** may also be connected to a remote network **50** (e.g., the Internet) or a remote site (e.g., a satellite, which is not shown in FIG. **1**). The remote network **50** or remote site allows the service center **20** to provide a wider variety of computer software, content, etc. that could be stored at. the service center **20**. The one or more databases **24** connected to the service center computer(s), e.g., computer $\mathbf{26}_1$, are used to store database entries consisting of computer software available on the computer(s) **26**. In one embodiment, each user computer $\mathbf{40}_1$–$\mathbf{40}_N$ has its own secure database (not shown), that is not accessible by any other computer. The communication links $\mathbf{30}_1$–$\mathbf{30}_N$ allow the one or more user computer systems $\mathbf{40}_1$–$\mathbf{40}_N$ to simultaneously connect to the computer(s) $\mathbf{26}_1$–$\mathbf{26}_M$. The connections are managed by the server **22**.

After a user computer system **40** establishes two-way communications with the information service computer **26**, the content is sent to the user computer system **40** in a manner hereinafter described. The downloaded content includes an application that surveys the user and/or the user computer system's hardware and/or software to develop a user profile as well as a profile of the user's system. The information gathered from the user and/or user's computer system is subsequently provided to the service center **20**, which provides additional content to the user computer **40** based on the user and system profile. The database entries from the database connected to the service computer **26** contain information about computer software, hardware, and third party services and products that are available to a user. Based on the user and/ or system profile, the content is further sent to the user computer for display. The content may also include a summary of information such as the availability of patches and fixes for existing computer software, new versions of existing computer software, brand new computer software, new help files, etc. The content may further include information regarding availability of hardware and third party products and services that is of interest to the user. The user is then able to make one or more choices from the summary of available products and services, and request that the products be transferred from the service computer **26** to the user computer. Alternatively, the user may purchase the desired product or service from the summary of available products and services.

FIG. **2** illustrates an exemplary computer system **100** that implements embodiments of the present invention. The computer system **100** illustrates one embodiment of user computer systems $\mathbf{40}_1$–$\mathbf{40}_N$ and/or computers $\mathbf{26}_1$–$\mathbf{26}_M$ (FIG. **1**), although other embodiments may be readily used.

Referring to FIG. **2**, the computer system **100** comprises a processor or a central processing unit (CPU) **104**. The illustrated CPU **104** includes an Arithmetic Logic Unit (ALU) for performing computations, a collection of registers for temporary storage of data and instructions, and a control unit for controlling operation for the system **100**. In one embodiment, the CPU **104** includes any one of the x86, Pentium™, Pentium II™, and Pentium Pro™ microprocessors as marketed by Intel™ Corporation, the K-6 microprocessor as marketed by AMD™, or the 6x86MX microprocessor as marketed by Cyrix™ Corp. Further examples include the Alpha™ processor as marketed by Digital Equipment Corporation™, the 680X0 processor as marketed by Motorola™; or the Power PC™ processor as marketed by IBM™. In addition, any of a variety of other processors, including those from Sun Microsystems, MIPS, IBM, Motorola, NEC, Cyrix, AMD, Nexgen and others may be used for implementing CPU **104**. The CPU **104** is not limited to microprocessor but may take on other forms such as microcontrollers, digital signal processors, reduced instruction set computers (RISC), application specific integrated circuits, and the like. Although shown with one CPU **104**, computer system **100** may alternatively include multiple processing units.

The CPU **104** is coupled to a bus controller **112** by way of a CPU bus **108**. The bus controller **112** includes a memory controller **116** integrated therein, though the memory controller **116** may be external to the bus controller **112**. The memory controller **116** provides an interface for access by the CPU **104** or other devices to system memory **124** via memory bus **120**. In one embodiment, the system memory **124** includes synchronous dynamic random access memory (SDRAM). System memory **124** may optionally include any additional or alternative high speed memory device or memory circuitry. The bus controller **112** is coupled to a system bus **128** that may be a peripheral component interconnect (PCI) bus, Industry Standard Architecture (ISA) bus, etc. Coupled to the system bus **128** are a graphics controller, a graphics engine or a video controller **132**, a mass storage device **152**, a communication interface device **156**, one or more input/output (I/O) devices $\mathbf{168}_1$–$\mathbf{168}_N$, and an expansion bus controller **172**. The video controller **132** is coupled to a video memory **136** (e.g., 8 Megabytes) and video BIOS **140**, all of which may be integrated onto a single card or device, as designated by numeral **144**. The video memory **136** is used to contain display data for displaying information on the display screen **148**, and the video BIOS **140** includes code and video services for controlling the video controller **132**. In another embodiment, the video controller **132** is coupled to the CPU **104** through an Advanced Graphics Port (AGP) bus.

The mass storage device **152** includes (but is not limited to) a hard disk, floppy disk, CD-ROM, DVD-ROM, tape, high density floppy, high capacity removable media, low capacity removable media, solid state memory device, etc., and combinations thereof. The mass storage device **152** may include any other mass storage medium. The communication interface device **156** includes a network card, a modem interface, etc. for accessing network **164** via communications link **160**. The I/O devices $\mathbf{168}_1$–$\mathbf{168}_N$ include a keyboard, mouse, audio/sound card, printer, and the like. The I/O devices $\mathbf{168}_1$–$\mathbf{168}_N$ may be a disk drive, such as a compact disk drive, a digital disk drive, a tape drive, a zip drive, a jazz drive, a digital video disk (DVD) drive, a solid state memory device, a magneto-optical disk drive, a high density floppy drive, a high capacity removable media drive, a low capacity media device, and/or any combination thereof. The expansion bus controller **172** is coupled to non-volatile memory **175** which includes system firmware **176**. The system firmware **176** includes system BIOS (numeral **82**, FIG. **3**), which is for controlling, among other things, hardware devices in the computer system **100**. The system firmware **176** also includes ROM **180** and flash (or EEPROM) **184**. The expansion bus controller **172** is also coupled to expansion memory **188** having RAM, ROM, and/or flash memory (not shown). The system **100** may additionally include a memory module **190** that is coupled to the bus controller **112**. In one embodiment, the memory module **190** comprises a ROM **192** and flash (or EEPROM) **194**.

As is familiar to those skilled in the art, the computer system **100** further includes an operating system (OS) and at least one application program, which in one embodiment, are loaded into system memory **124** from mass storage device **152** and launched after POST. The OS may include any type of OS including, but not limited or restricted to, DOS, Windows™ (e.g., Windows 95™, Windows 98™, Windows NT™), Unix, Linux, OS/2, OS/9, Xenix, etc. The operating system is a set of one or more programs which control the computer system's operation and the allocation of resources. The application program is a set of one or more software programs that performs a task desired by the user.

In accordance with the practices of persons skilled in the art of computer programming, the present invention is described below with reference to symbolic representations of operations that are performed by computer system **100**, unless indicated otherwise. Such operations are sometimes referred to as being computerexecuted. It will be appreciated that operations that are symbolically represented include the manipulation by CPU **104** of electrical signals representing data bits and the maintenance of data bits at memory locations in system memory **124**, as well as other processing of signals. The memory locations where data bits are maintained are physical locations that have particular electrical, magnetic, optical, or organic properties corresponding to the data bits.

When implemented in software, the elements of the present invention are essentially the code segments to perform the necessary tasks. The program or code segments can be stored in a processor readable medium or transmitted by a computer data signal embodied in a carrier wave over a transmission medium or communication link. The "processor readable medium" may include any medium that can store or transfer information. Examples of the processor readable medium include an electronic circuit, a semiconductor memory device, a ROM, a flash memory, an erasable ROM (EROM), a floppy diskette, a CD-ROM, an optical disk, a hard disk, a fiber optic medium, a radio frequency (RF) link, etc. The computer data signal may include any signal that can propagate over a transmission medium such as electronic network channels, optical fibers, air, electromagnetic, RF links, etc. The code segments may be downloaded via computer networks such as the Internet, Intranet, etc.

FIG. **3** illustrates a logical diagram of computer system **100**. Referring to FIGS. **2** and **3**, the system firmware **176** includes software modules and data that are loaded into system memory **124** during POST and subsequently executed by the processor **104**. In one embodiment, the system firmware **176** includes a system BIOS module **82** having system BIOS handlers, hardware routines, etc., a ROM application program interface (RAPI) module **84**, an initial start-up application (ISUA) module **86**, an initial payload **88***a*, cryptographic keys **90**, a cryptographic engine **92**, and a display engine **94**. The aforementioned modules and portions of system firmware **176** may be contained in ROM **180** and/or flash **184**. Alternatively, the aforementioned modules and portions of system firmware **176** may be contained in ROM **190** and/or flash **194**. RAPI **84** provides a secure interface between ROM application programs and system BIOS **82**. The RAPI **84**, ISUA **86**, and initial payload **88***a* may each be separately developed and stored in the system firmware **176** prior to initial use of the computer system **100**. In one embodiment, the RAPI **84**, ISUA **86**, and initial payload **88***a* each includes proprietary software developed by Phoenix Technologies, Ltd. One embodiment of RAPI.**84** is described in copending U.S. patent application Ser. No. 09/336,889 entitled "System and Method for Securely Utilizing Basic Input and Output System (BIOS) Services," filed on Jun. 18, 1999, assigned to Phoenix Technologies, Ltd., and which is incorporated herein by reference. One embodiment of ISUA **86** is described in co-pending U.S. patent application Ser. No. 09/336,289 entitled "Method and Apparatus for Automatically Installing and Configuring Software on a Computer," filed on Jun. 18, 1999, assigned to Phoenix Technologies, Ltd., and which is incorporated herein by reference.

In one embodiment, as shown in FIGS. **3** and **4A** and **4B**, after power is initially turned on to a new computer system **100**, the system commences with POST procedures. During the initial POST, the ISUA **86** is transferred to the mass storage device **152**, as shown by **A1**. In one embodiment, such a transfer is made during the manufacturing and/or assembly process, when the system **100** is first powered up after the operating system has been installed (but prior to loading and running the operating system). In an alternative embodiment, such a transfer may be made after the manufacturing and/or assembly process, after the user receives and powers up the system **100**. In a further alternate embodiment, during the transfer of the ISUA **86**, additional programs, applications, drivers, data, graphics and other information may also be transferred (for example, from ROM) to the mass storage device **152**. For example, the transfer may include the transfer of the initial payload **88***a* to the mass storage device **152**, subsequent to which the initial payload is delivered from the mass storage device **152**. Alternatively, the initial payload may be delivered from the ROM. One embodiment of the system and process for facilitating such a transfer is described in co-pending U.S. patent application Ser. No. 09/336,067 entitled "System and Method for Transferring an Application Program from System Firmware to a Storage Device" filed on Jun. 18, 1999, which is assigned to Phoenix Technologies, Ltd., the contents of which are incorporated herein by reference. Alternative embodiments of the system and process for facilitating such a transfer are described in co-pending U.S. patent application Ser. No. 09/272,859, entitled "Method and Apparatus for Providing Memory-based Device Emulation" filed on Mar. 19, 1999, in co-pending U.S. Patent Continuation-in-Part application Ser. No. 09/336,307 entitled "Method and Apparatus for Providing Memory-Based Device Emulation" filed on Jun. 18, 1999, and in co-pending U.S. patent application Ser. No. 09/336,281 entitled "System and Method for Inserting One or More Files Onto Mass Storage" filed Jun. 18,1999, each of which is assigned to Phoenix Technologies, Ltd., the assignee of the present invention, the contents of each of which are incorporated herein by reference.

In one embodiment, the ISUA **86** is a computer software executable program that will determine if there are preinstalled programs that are resident on the end user's system. If so, it will identify those preinstalled programs and create shortcuts (on the desktop in the case of a Windows operating system), or bookmarks, to allow the user to automatically launch the programs. In this embodiment, the executable program is also capable of initiating and establishing two-way communications with one or more applications on the server **22** and/ or any one of the service computers **26** (FIG. **1**), as described below. Moreover, in one embodiment, graphical content of the initial payload **88***a* is displayed by display engine **94** on the user's display screen **148** during POST. Alternatively, the graphical content of the initial payload **88***a* may be displayed after a subsequent booting process. For example, as part of the user's profile as

described below, the user may be asked if he or she would like to obtain additional information regarding one or more products and/or services. If the user so desires, content regarding the desired products and/or services will be displayed during subsequent boot processes.

Once POST is completed, the OS is loaded, executed, and initialized. Standard OS drivers and services are then loaded. The user is then prompted to enter registration information including demographic information such as age, gender, hobbies, etc. In addition, the ISUA **86** is executed, and runs in the background, remaining idle until it detects a communication link established between the computer system **100** and a remote server (e.g., server **22** of FIG. **1**) over Network **164** of FIG. **2** (e.g., over the Internet). In one embodiment, the ISUA 86 may search through the operating system to determine if there are applications that have been pre-loaded and pre-installed onto the system. If so, the ISUA 86 may automatically provide short cuts and/or bookmarks for the applications to launch into a predetermined server once the communication link is established. This communication link can be established with a network protocol stack, (e.g. TCP/IP) through sockets, or any other two-way communications technique known in the art. Once the communication link **30** is established, the ISUA 86 issues a request signal to the server **22** (as shown by A2) to download an initial content package **62** from a content module **60**. Responsive to the request, the server downloads the initial content package **62** (as shown by A3), which, in one embodiment, is stored in the mass storage device **152**. In one embodiment, the initial content **62** and subsequent content **64** may be developed separately, and each is encrypted and/or digitally signed using encryption keys, prior to storing of the initial content **62** and subsequent content **64** on the server **22**. When the initial content **62** and/or subsequent content **64** is/are subsequently downloaded into system **100**, the crypto engine **92** will use keys **90** to decrypt the initial content **62** and/or subsequent content **64**.

As discussed earlier, the initial content package **62** may include applications **62***a*, drivers **62***b*, and payloads **62***c*. In one embodiment, the applications **62***a* include a data loader application and a profile manager application. The data loader application functions in the same or a similar manner as ISUA **86**, and once downloaded, disables and replaces the ISUA 86. More specifically, the data loader application is a computer software program which is also capable of initiating, establishing, and terminating two-way communications between the server **22** and the computer system **100**. The data loader application also provides traffic control management between the server **22** and computer system **100**, as well as other functions to facilitate communication between the end user's system and the designated server, and content downloading to the end user's system.

The profile manager obtains the user and system profiles of the computer system **100** based on user preferences, system hardware, and software installed at the computer system **100**. Upon obtaining the user and system profile of the computer system **100**, the profile manager application forwards the results to the data loader application, which subsequently provides the information to the server **22**, which matches the user indicted preferences with database **24** (FIG. **1**). The results may be forwarded at predetermined intervals or at the user's request. The server **22** then processes the user profile or demographic data and targets content to the users which have similar profiles. In addition, the user profile data of a plurality of users are compiled on the server **22** and aggregated to create an aggregate user profile model. Content is then transmitted to user computer system's based on the user profile data and/or the aggregate user profile model (as shown-by A4). The subsequent content **64** is downloaded and stored in system firmware **176**, designated by numeral **88***b*. In one embodiment, the subsequent content **64** is stored in non-volatile memory such as flash or EEPROM, with the loading of the subsequent content being done by refashion the ROM, as is well known by those skilled in the art. The subsequent content **64** may also be stored as one or more files on mass storage device **152** or may be used to modify the Windows™ system file (under the Windows™ environment). The profile collection process is continued as long as the computer system **100** is activated. In one embodiment, content may be downloaded after the user's profile is received and analyzed at the server **22**.

When the computer system **100** is subsequently powered up (see FIG. **4B**), the system again performs POST. The content that was previously downloaded and stored in system firmware **176**, and subject to copyright issues being resolved, is then displayed, prior to loading and/or execution of the operating system. In the Windows™ environment, the Windows™ logo, which is displayed during the initial loading of the operating system, is subsequently replaced by one or more screen that display the previously downloaded content stored in system firmware **176**.

In the case of storing the content as one or more files on the mass storage device **152**, as opposed to reflashing the ROM, the Windows™ logo file, which is displayed during boot-up and shutdown, may be altered or replaced. One embodiment utilizing this approach involves replacing the corresponding Windows™ system files with the one or more files showing the content (e.g., graphic file), as described in co-pending U.S. patent application Ser. No. 09/336,003 entitled "Displaying Images during Boot-up and Shutdown" filed on Jun. 18, 1999, which is assigned to Phoenix Technologies, LTD., the contents of which are incorporated herein by reference. The boot-up Windows display file is named LOGO.SYS and is usually located in the Windows directory. First the Windows™ LOGO.SYS file is transferred from the Windows directory to another directory. Then, the content graphics file is renamed as LOGO.SYS and is transferred to the Windows™ directory. The operating system retrieves this file when the operating system is first launched, and hence the content is displayed on the display screen. Windows™ expects the LOGO.SYS file to be a bit-mapped file with resolution 320x400 and 256 colors although Windows™ will later stretch the resolution to 640x400 for displaying purposes. Therefore, the content graphics file is to be the same graphics format (usually named with the extension ".BMP" before being renamed to LOGO.SYS).

The operating system is then loaded, executed, and initialized. The standard operating system drivers and applications are also loaded. The profile manager is then executed. When a link has been established with the predetermined web site, additional content may be downloaded and subsequently displayed. Such additional content are either provided arbitrarily or provided based on the information obtained from a survey of the user or the user's system. In one embodiment, once the boot process is completed, a portion of the display screen may bemused to provide icons or shortcuts that are used to access detailed information regarding the previously displayed messages with graphics or informational material. In a further embodiment, the messages with graphics or informational material may again be displayed during the shut-down process, for example, replacing the screen display that displays the message "Win-

dows is shutting down" or "It is now safe to turn off your computer" with other selected content.

1. Boot Frequency

FIG. **5** shows an exemplary flow diagram of a portion of a BIOS profile module **200**, according to one embodiment of the present invention. The BIOS profile module **200** is part of the system BIOS **82** (FIG. **2**), or is an add-on to BIOS **82**. Initially when the computer system **100** (FIG. **2**) is first powered on, the BIOS profile module **200** sets up a BOOT_COUNT and initializes it to zero, either on mass storage **152** or flash memory **184** (e.g., some form of non-volatile storage). The BOOT_COUNT tracks the number of system boots either as a raw number or on a frequency basis (e.g., monthly, weekly, etc.). The BIOS profile module **200** is executed at some point during BIOS POST, as shown by block **205**. In one embodiment, the BIOS profile module **200** is executed at the end of BIOS POST, just before the operating system is launched.

Referring to FIG. **5**, the module **200** retrieves the system time and date (block **210**), which is read from the system clock/timer chip. At block **215**, it is determined whether a predetermined time interval has expired. This involves reading a previous time and date, subtracting the current time and date from the previous time and date to provide a result, and comparing the result with the predetermined time interval (e.g., yearly, monthly, weekly, daily, etc.). The previous time and date and predetermined time interval are stored in non-volatile memory. Initially, when the computer system **100** is first powered up, an initial previous time and date are stored. If the time interval has not expired, then BOOT_COUNT is incremented (block **220**) and module **200** ends.

However, if the time interval has expired, then BOOT_COUNT and result are stored in a separate non-volatile memory location (block **225**) for later transmission to, for example, profile server **22** (FIG. **1**). In addition, a frequency based on the BOOT_COUNT and result is calculated (block **230**) indicating an average of how often the computer system **100** is booted. The frequency is stored with BOOT_COUNT and result, as a single record. A number of such records are created over time, and the one or more records are transmitted to profile server **22** periodically. The one or more records are then cleared. Continuing to refer to FIG. **5**, the current time and date are stored as the previous time and date (block **235**), BOOT_COUNT is cleared (block **240**), and the module **200** ends.

The boot frequency is used by the profile server **22** to determining how often the user is on the computer system **100**, and hence the effectiveness -of content delivered to the computer system **100**. In addition, the boot frequency is a factor in determining the periodicity for delivering content to computer system **100**.

2. Boot Duration

FIG. **6** shows a flow diagram showing a process **250** for determining boot duration, according to one embodiment of the present invention. Referring to FIG. **6**, the process **250** commences during BIOS POST by, for example, the system BIOS **82** (FIG. **2**). Immediately after enabling the video display during BIOS POST (block **255**), the system time and date are read and stored in a temporary location (e.g., system memory **124**, mass storage **152**, etc. of FIG. **2**), as shown in block **260**. Reading the system time and date generally involves reading an I/O address where the system timer registers reside.

At the end of BIOS POST, the operating system is loaded, and, at the end of the loading of the operating system, an OS profile module is loaded (block **265**). In one embodiment, the OS profile module is one of the application programs **62***a* or drivers **62***b* shown in FIG. **3**, and is loaded at startup of the operating system. Once the OS profile module is loaded, it is assumed that the operating system or user desktop is available. Once loaded, the OS profile module reads the system time and date (block **270**). The OS profile module then subtracts the first time and date located in the temporary location from the second time and date (block **275**). The result is the boot duration, i.e., the time that it takes to load the operating system. The boot duration is then stored as a record in non-volatile memory (e.g., mass storage **152**, flash **184**, etc.), as shown in block **280**.

In one embodiment, each time the computer system **100** is booted, the boot duration is stored as a record. Alternatively, the boot duration is calculated and stored periodically (e.g., every week, month, etc.). It is important to calculate the boot duration each time or periodically because the boot duration may vary from one boot to another. This could be attributed to a number of factors, including, for example, the changing of machine settings (e.g., adding or removing hardware such as DVD-ROM drive), memory chip problems during BIOS POST which causes a change in the amount of detected memory, power fluctuations which cause hard disk spin up times to vary, and the like.

The one or more records collected are periodically transmitted to profile server **22** (FIG. **1**). The boot duration indicates to the profile server **22** the amount of time available for content to be displayed. The more time content is displayed for end users, the more the originators of the content can be charged for displaying such content. A similar process can be used to determine the shut-down time.

3. Internet Connection Frequency

FIG. **7** shows an exemplary flow diagram of a portion of the OS profile module **300**, according to one embodiment of the present invention. In one embodiment, the OS profile module **300** is an application program **62***a* or driver **62***b* stored on mass storage **152** (FIG. **3**). Initially when the OS profile module **300** is first executed, the OS profile module **300** sets up a CONNECTION_COUNT and initializes it to zero, either on mass storage **152** or flash memory **184** of FIG. **2** (e.g., some form of non-volatile storage). The CONNECTION_COUNT tracks the number of connections to the Internet or other network either as a raw number or on a frequency basis (e.g., monthly, weekly, etc.). The OS module **300** watches for an Internet connection while running in the background. For a description of embodiments showing how the Internet connection detection operates, see co-pending U.S. patent application Ser. Nos. 09/336,108 and 09/336,289 entitled "Method and Apparatus for Creating and Deploying Smaller Microsoft Windows Applications for Automatic Configuration of a Computing Device" and "Method and Apparatus for Automatically Installing and Configuring Software on a Computer", filed concurrently herewith and assigned to Phoenix Technologies Ltd., the assignee of the present invention, the contents of which are herein incorporated by reference. When a connection is detected (block **305**), the module **300** increments CONNECTION_COUNT (block **310**) and stores it back to memory.

At block **315**,. the module **300** retrieves the current system time and date, which are read from the system clock/timer chip. At block **320**, it is determined whether a predetermined time interval has expired. This involves reading a

previous time and date, subtracting the current time and date from the previous time and date to provide a result, and comparing the result with a predetermined time interval (e.g., yearly, monthly, weekly, daily, etc.). The previous time and date and predetermined time interval are stored in non-volatile memory. Initially, when the OS profile module **300** is first powered up, an initial previous time and date is stored. If the time interval has not expired, then the module ends.

However, if the time interval has expired, CONNECTION_COUNT and result are stored in a separate non-volatile memory location (block **325**) for later transmission to, for example, profile server **22** (FIG. **1**). In addition, a frequency, based on the CONNECTION_COUNT and result, is calculated (block **330**) indicating an average of how often there is an Internet connection. The frequency is stored with CONNECTION_COUNT and result, as a single record. A number of such records are created over time, and are transmitted to profile server **22** periodically. The one or more records are then cleared. The current time and date are stored as the previous time and date (block **335**), CONNECTION_COUNT is cleared (block **340**), and the module **300** ends.

The connection frequency is used by the profile server **22** to determining how often the user is connected to the Internet, and hence the amount of and periodicity of content that can be downloaded to computer system **100**.

4. Internet Connection Duration

FIG. **8** shows an exemplary flow diagram of another portion of the OS profile module **400**, according to one embodiment of the present invention. Referring to FIG. **8**, the OS profile module **400** is loaded after the operating system is loaded (block **405**). The OS profile module **400** then determines whether it is in a failsafe mode by interrogating a failsafe flag. The failsafe flag, which is optionally user alterable, is stored in non-volatile memory or is hard-coded as part of the module **400**. In the case where failsafe mode is not used, the OS profile module **400** continuously monitors for an Internet connection while running in the background (block **415**). When a connection is detected, the OS profile module **400** reads the system time and date, and stores the same in non-volatile memory as connection start_time. In the Windows™ environment, the module **400** uses operating system call GetSystemTime( ) to retrieve the system time and date.

When the computer system **100** (FIG. **2**) is disconnected from the Internet, the module **400** detects the disconnection (block **425**), retrieves the time and date (e.g., using the GetSystemTime( ) call), as shown in block **430**, and temporarily saves the same (e.g., in RAM) as connection end_time. The connection start_time is subtracted from the connection end_time to determine a connection_duration (block **435**). The connection duration is stored in a record in non-volatile memory-for later delivery to the profile server **22** (FIG. **1**). At block **485**, a FAIL_FLAG is reset to zero (used in failsafe mode), and is described in the following paragraphs.

In an alternative implementation, the failsafe mode is utilized. The failsafe mode takes into account the fact that the computer system **100** may shut-off abruptly, and recovers in a well-behaved manner. In the failsafe mode, the OS profile module **400** utilizes an operating system SetTimer( ) service to get a periodic signal from the operating system. When the module **400** receives the signal from the operating system, the module verifies that the computer system **100** is still connected, and, if so, retrieves the time and date using GetSystemTime( ) and stores the time and date in non-volatile memory as connection tally time. This helps overcome situations where the computer system **100** loses power or is powered off without first disconnecting from the Internet.

Continuing to refer to FIG. **8**, the OS profile module **400** determines whether the prior connection ended properly (block **450**). This is accomplished by using a FAIL_FLAG that is stored in non-volatile memory. When the module **400** is started for the first time, FAIL_FLAG is set to zero. If FAIL_FLAG is modified to one and is not cleared, indicating that the prior connection did not end properly, the connection_duration is calculated using connection start_time and connection tally-time stored in non-volatile memory. The connection duration is then stored in non-volatile memory (block **455**). On the other hand, if FAIL_FLAG is equal to zero, then the module **400** moves to block **460**. At block **460**, the module **400** again waits for an Internet connection. Once an Internet connection is detected, the time and date are retrieved and saved as connection start-time, and FAIL_FLAG is set to one. At block **465**, the periodic notification is setup using the operating system SetTimer( ) service. At block **470**, a notification is received, and at block **475** it is determined whether there is still an Internet connection. If there is an Internet connection, the time and date are retrieved and saved as connection tally_time in non-volatile memory (block **480**). Blocks **470**, **475**, and **480** are continuously executed until there is no longer an Internet connection, at which time blocks **430**, **435**, and **485** are executed as before, and the module ends.

The connection duration together with the connection frequency provide the profile server **22** with the ability to determining how often and for how long the user is connected to the Internet or other network.

5. System Usage Duration

FIG. **9** shows an exemplary flow diagram of yet another portion of the OS profile module **500**, according to one embodiment of the present invention. Referring to FIG. **9**, the OS profile module **500** is loaded after the operating system is loaded (block **505**). The OS profile module **500** then determines whether it is in the failsafe mode. If failsafe mode is not used, the OS profile module **500** retrieves the system time and date (e.g., using the GetSystemTime( ) call) and saves the same as usage start_time in non-volatile memory (block **515**). When a shutdown of computer system **100** (FIG. **2**) is detected (block **520**), the module **500** retrieves the time and date again (block **525**), and temporarily saves the same as usage end_time. The usage start_time is subtracted from the usage end_time to determine the usage-duration (block **530**). The usage duration is stored in a record in non-volatile memory for later delivery to the profile server **22** (FIG. **1**). At block **535**, a USAGE_FLAG is reset to zero (used in failsafe mode), as will be described in the following paragraphs.

In the failsafe mode, the OS profile module **500** determines whether the prior usage ended properly (block **550**). This is accomplished by using the USAGE_FLAG, which is stored in non-volatile memory. When the module **500** is started for the first time, USAGE_FLAG is set to zero. If USAGE_FLAG is modified to one and is not reset, indicating that the prior usage of computer system **100** did not end properly, the usage_duration is calculated using usage start_time and usage tally_time stored in non-volatile memory. Then usage_duration is stored as a record in

non-volatile memory (block **555**). On the other hand, if USAGE_FLAG is equal to zero, the module **500** retrieves and stores the system time and date as usage start_time, and USAGE_FLAG is set to one (block **560**). At block **565**, the periodic notification is setup using the operating system SetTimer( ) service. At block **570**, a notification is received, and at block **575** it is determined whether a shutdown is detected. If a shutdown is not detected, then the time and date are retrieved and saved as usage tally_time in non-volatile memory (block **580**). Blocks **570**, **575**, and **580** are continuously executed until a shutdown is detected, at which time blocks **525**, **530**, and **535** are executed as before, and the module ends.

The system usage duration provides the profile server **22** information on how long the user is on computer system **100**.

Thus, as can be seen, the present invention provides numerous advantages, some of which include the ability to determine computer system usage profile, which is periodically provided to a server, and the ability to intelligently target content from the server to the computer system responsive to the computer system usage profile. The intelligent decisions include, for example, the amount of and periodicity of content to download, the price to charge third-party content providers for the content, and the like, in order to maximize the efficiency and effectiveness of the content.

While certain exemplary embodiments have been described and shown in the accompanying drawings, it is to be understood that such embodiments are merely illustrative of and not restrictive on the broad invention, and that this invention not be limited to the specific constructions and arrangements shown and described, since various other modifications may occur to those ordinarily skilled in the art.

What is claimed is:

1. A method of determining boot-up time of a computer system, comprising:

retrieving a first time on the computer system;

loading an operating system;

retrieving a second time on the computer system when the operating system has loaded;

determining a boot time in response to the first and second times; and

determining an amount of content to transfer to the computer system based on the boot time.

2. The method of claim **1** further comprising transferring the boot time to a server.

3. The method of claim **2** further comprising transferring an amount of content responsive to the boot time.

4. The method of claim **2** further comprising adjusting the periodicity of the content to be transferred responsive to the boot time.

5. The method of claim **4** further comprising transferring the content from the server to the computer system.

6. The method of claim **1** further comprising storing each boot time as a record.

7. The method of claim **1**, wherein retrieving a first time comprises retrieving the first time from the system timer of the computer system using a BIOS profile module.

8. The method of claim **1**, wherein retrieving a second time comprises e second time from the system timer on the computer system once the operating oaded using an OS profile module, said OS profile module to be loaded once the stem has been loaded.

9. The method of claim **4**, further comprising determining a boot the computer system, said boot frequency to be used in said adjusting the periodicity of the content to be transferred responsive to the boot time.

10. A computer program product, comprising:

a computer usable medium having computer program code embodied therein to retrieve a first time on the computer system;

computer readable program code to cause an operating system to load;

computer readable program code to retrieve a second time on the computer system when the operating system has loaded;

computer readable program code to determine a boot time in response to the first and second times; and

computer readable program code to determine an amount of content to transfer to the computer system based on the boot time.

11. The computer program product of claim **10** further comprising computer readable program code to transfer the boot time to a server.

12. The computer program product of claim **11** further comprising computer readable program code to transfer an amount of content responsive to the boot time.

13. The computer program product of claim **11** further comprising computer readable program code to adjust the periodicity of the content to be transferred responsive to the boot time.

14. The computer program product of claim **13** further comprising computer readable program code to transfer the content from the server to the computer system.

15. A system, comprising:

a computer including,

a memory element having one or more instructions, and

a processor coupled to the memory element, the processor, in response to the one or more instructions to,

retrieve a first time on the computer,

load an operating system,

retrieve a second time on the computer when the operating system has loaded,

determine a boot time in response to the first and second times; and

transmit the boot time when a network connection is detected; and

a server to receive the boot time, determine an amount of content to transfer to the computer based on the boot time, transmit said amount of content to the computer, and charge said amount from a source of the content responsive to the boot time.

16. The system of claim **15** wherein the server to adjust an amount of content to transfer responsive to the boot time.

17. The system of claim **15** wherein the processor to transmit a frequency of boots of the computer to the server such that the server adjusts a periodicity of the content that is transferred to the computer.

* * * * *